64 PLUS 4



& AMIGA

WRZESIEŃ 1991 ISSN 0867-3918 INDEKS 377112 CENA 8.000 zł

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE

BUK

KSIĄŻKA „C-64 KOD MASZYNOWY", cena 30.000zł. Płatne przy odbiorze. Rafał Zimowski, Łódź, ul. Bratysławska 14 m.11.

Grupa ASTEC nawiąże kontakt z ambitnymi i zdolnymi programistami C-64. Robert Kuna, ul. Paderewskiego 3/12, 63-102 ŚREM.

Sprzedam dyski nagrane na AMIGĘ. Paweł Lewandowski, ul. Chałubińskiego 5c/5, 67-100 NOWA SÓL.

Wymienię się chętnie grami lub kupię je (C-64, magnetofon). Kubyszek Bartosz, ul. Tuwima 5/11, 43-300 Bielsko Biała, tel. 454-62.

Wymienię programy na Amigę 500. Zbyszek Kułakowski, ul. Słowicz 4, 22-400 ZAMOŚĆ.

SPRZEDAM: C-64, wszystkie peryferia, 150 dyskietek, literatura, etc. Komplet lub niektóre elementy. Tel. KRAKÓW 36-57-08.

WYMIENIĘ PROGRAMY C-64 (KASETA). Michał Groth, ul. Pomorska 22A/20, 80-333 Gdańsk.

C-64+stacja 1541II+magnetofon; wymienię literaturę i oprogramowanie. Tomasz Zarzycki, 59-255 Tomaszów Bol. 130, woj. legnickie.

TANIO SPRZEDAM AMIGĘ 500 + OSPRZĘT + OPROGRAMOWANIE. Bielsko-Biała, tel. 438-49.

Klub korespondencyjny THE BLOODY GREMLINS zacznie wymieniać oprogramowanie z AMIGOWCAMI. Adres: The Bloody Gremlins - Wójcicki Kamil, ul. Nowogrodzka 60, 95-200 Pabianice, tel. 15-02-60.

POSZUKUJĘ PROGRAMÓW 3D NA AMIGĘ. Tel. 520-335, ul. Kościuszki 112/4, Poznań.

GRY I PROGRAMY NA C-64. KATALOG - DARMO. Noskowskiego 12/21, Jelenia Góra.

OGŁOSZENIA

KOMPUTEROWA FIRMA USŁUGOWA "TREND" - COMMODORE AMIGA: oprogramowanie, literatura. Informacja : dyskietka lub koperta + znaczek. Kontakt: Rafał Wierzbicki, ul. Rogowska 86/10, 54-440 Wrocław.

THE MANY AMI nawiąże kontakt z amigowcami. Kuba Kopczyk, Os. Wichrowe Wzg. 26/73, 61-697 Poznań.



miesięcznik nr 9(11) wrzesień 1991
cena 1 egz.: 8000 zł



Wydawca:
ABUK Spółka z o. o.

Redakcja nie ponosi odpowiedzialności za treść ogłoszeń.

Adres redakcji: Redakcja „64 plus 4" 85-166 Bydgoszcz 43 skrytka pocztowa 64

Redagują: Marcin Dudar, Sambor Kuźma, Paweł Sołtysiński, Waldemar Szczygieł (red. nacz.), Krzysztof Kobuz.

Okładka: Piotr Bartz.

Skład: ABUK

Druk: Z.P. POLRASTER 85-353 Bydgoszcz, ul. Orawska 19.

OD REDAKCJI

Wadliwa dystrybucja „64 plus 4 & Amiga" przez przedsiębiorstwo RUCH jest przyczyną kłopotów, jakie mają czytelnicy chcący nabyć nasze pismo. Zdarza się, że otrzymujemy jako zwroty NIETKNIĘTE paczki zbiorcze. Tymczasem czytelnicy sygnalizują, że do wielu kiosków nie dociera ono wcale. Dlatego:

zapraszamy wszystkich chętnych do prowadzenia kolportażu „64 plus 4 & Amigą" (kluby, studia i sklepy komputerowe, księgarnie, osoby indywidualne itd.) do współpracy!

Oferujemy korzystne warunki!

Zainteresowanych prosimy o kontakt z działem dystrybucji pod adresem: Przedsiębiorstwo ABUK, 87-200 Wąbrzeźno, ul. 1 Maja 33.

NOWOŚĆ!

PUBLIC DOMAIN PACK dla COMMODORE 64 NA KASECIE!

Jeszcze w tym roku ukażą się cztery kasety!

Na naszych zestawach znajdują się programy, które do tej pory ukazały się na dyskach PDP, oraz wiele nowości!

Pojedyńcza kaseta kosztuje 30.000zł, wykupienie prenumeraty do końca roku (4 kasety) kosztuje tylko 110tys zł.

Wśród wszystkich, którzy zamówią komplet kaset (blankiet wpłaty zamieszczamy na str. 15) rozlosujemy 10 bezpłatnych prenumerat naszego pisma na rok 1992! Na blankiecie prosimy dopisać PDP-taśma. Spis treści pierwszego zestawu na str. 14.

Przedsiębiorstwo ABUK S-ka z o.o. oferuje państwu **szybką i tanią obsługę reklamową.** Ogłoszenia drobne od osób indywidualnych (do 10 słów) przyjmujemy bezpłatnie. Większe - 1000 zł za słowo. Reklamy ramkowe (minimalny format - 20 cm^2): $1cm^2$ ogłoszenia-4500zł, cała strona - 2,5 mln zł; kolor - odpowiednio 100% drożej.

Ogłoszenia przyjmujemy za pośrednictwem poczty (nasz adres - patrz stopka redakcyjna). Treść ogłoszenia z określeniem formatu reklamy (ewentualnie zamówieniem koloru) prosimy nadsyłać listem poleconym wraz z odcinkiem wpłaty (za pomocą przekazu pieniężnego) na konto Przedsiębiorstwa ABUK Bank Polska Kasa Opieki SA Oddział w Bydgoszczy, konto nr : 5.09011-400522.7-136-11-111.0 Dołączenie do zamówienia odcinka wpłaty przyspieszy zamieszczenie reklamy .

W numerze :

W następnym numerze:

- **Przedstawimy mini skaner do C-64**
- **AMIGA - język C**
- **Cheats na C-64 i Amigę**

Z daleka i z bliska

- Termomagnetyczne dyski optyczne robią ostatnio furorę wśród użytkowników komputerów.

Zasada działania tych dysków jest prosta: zapis następuje poprzez podgrzanie warstwy nośnika promieniem lasera do temperatury ok. 400 stopni C, temperatura ta powoduje „bałagan"w dipolach magnetycznych nośnika. Podgrzany obszar stygnie w jednorodnym polu magnetycznym, przyjmując jego polaryzację. Odczyt następuje analogicznie, jak w popularnych urządzeniach CD.

Wiadomości te, przydadzą się podczas lektury poniższego tekstu, opisującego dwa typy dysków optycznych do Amigi: Ricoh Optical OD-600 i Sony Optical SM-S501-11.

Oba wymienione urządzenia są ze sobą kompatybilne, tj. dyski zapisane na jednym z nich dają się bez problemu odczytywać na drugim. Dyski te są dostarczane z dyskietkami (?) o pojemności około 600 MB. Cena takiej dyskietki wynosi około 200 marek i daje w przeliczeniu koszt ok.0.33 marki za jeden megabajt pamięci. Cena stacji Ricoh wynosi (wraz z dyskietką) ok. 9000 marek, Sony (też z dyskietką) ok. 6500. Poniżej znajdują sie parametry techniczne tych stacji w porównaniu do zainstalowanego w A3000 standardowego dysku twardego Quantum 105LPS.

Oto one:

	Ricoh	Sony	Quantum
Create (Files/s)	124	192	192
Open/Close (Files/s)	32	58	192
Scan (Files/s)	34	60	305
Delete (Files/s)	308	632	578
Seek/Read	667	752	942
Create (KByte/s)	89	192	656
Write (KByte/s)	117	262	921
Read (KByte/s)	358	401	910

Z parametrów i ceny należałoby polecać napęd Sony. Na zakończenie mała dygresja: licząc cenę stacji i jednej dyskietki otrzymujemy cenę jednego megabajta nastepującą: Ricoh - 15 marek, Sony 10.8 marek. Mogę tylko dodać, że MacWorld 8/91 reklamuje kasowalny dysk optyczny o pojemności.. 6.5 gigabajtów z twardym dyskiem w charakterze bufora w cenie 10.000 dolarów, co daje koszt 1 MB na poziomie... 2.6 marki.

- Nowy emulator Atari ST nazywa się Chamaleon II i oferowany jest z oryginalnym Atari-Tos-Romem. Karty 68020/030, o ile są, mogą działać pod TOSem 1.0, 1.2 i 1.4. Wszystkie rozdzielczości ST są osiągalne, a software'owy flicker-fixer (teoretycznie) zmniejsza dokuczliwość interlace'u. Chętni mają możliwość emulacji do ośmiu ST. Poza tym mamy możliwość zapisu i odczytu dysków w formacie ST, korzystania z amigoskich dysków twardych i botowania (niech ktoś wymyśli jakieś rozsądne słowo zamiast „boot") bezpośrednio z partycji ST.
- Firma California Access oferuje do A2000 kontroler SCSI Malibu Board w cenie ok. 400 marek. Napęd może być zainstalowany na płycie kontrolera. Oprócz tego jest tam miejsce na 8 MB RAM'u (moduły SIMM). Kontroler zapewnia 16-bitową transmisję danych przy szybkości (z napędem Quantum) około 700 KB/s. Od wersji Kikstarta 1.3 kontroler pozwala na autoboot.
- Ciekawostka: niemiecka firma Christian Obermaier wykonuje specyficzny typ usług: mozolnie policzone, lub narysowane obrazki z Amigi można przenieść na zwykły papier fotograficzny, lub diapozytyw. Grafiki mogą mieć rozdzielczość do 4000 na 4000 pikseli w formacie 24-bitowym. Zdjęcia mogą być formatu 24 na 36 mm, lub 18 na 24 cm. Cena naświetlenia jednego przezrocza 24 na 36 mm wynosi 30 marek.
- Czasopismo AMIGA ogłosiło konkurs, w którym nagrodą jest aparat fotograficzny Canon RC-260 pozwalający na zapis do 50 zdjęć na dyskietce o wymiarze 2 cali. Konkurs polega na tym, że należy przesłać własną grafikę na adres redakcji. Grafika ta, o ile będzie na odpowiednim poziomie, zostanie zamieszczona na pierwszej stronie okładki czasopisma.

A zatem - do dzieła!

- Na rynku pojawiło sie wiele kart graficznych likwidujących nieszczęsny amigowy interlace. Aby uławtić klientom zakup AMIGA 5/91 podaje testy poszczególnych typów kart. Oto wyniki testu (w skali od 0 do 12):

A2320	9.4
De-Interlace-Card	9.3
Highgraph V	9.1
FlickerFixer	7.9
X-tension Pro Video	9.0

Warto może dodać, że z reguły karty te oprócz usunięcia interlace'u dają możliwość pracy w trybie SuperHiRes.

Jarosław Chrostowski

Uczymy się grać

W poprzednim artykule przedstawiliśmy Wam prosty program narzędziowy, ułatwiający komponowanie własnych melodii, lub przenoszenie ich z partytur.

W dzisiejszym spotkaniu z muzyką dla C-16 i Plus 4 prezentujemy kolejny program tego typu, pozwalający tworzyć melodie w sposób mniej kłopotliwy. Będziecie poprostu grać na klawiaturze komputera, a w okienku uzyskacie nazwę nuty i wartość dźwięku.

Zanotowany przez Was na kartce ciąg tak uzyskanych wartości nut można utrwalić, przenosząc je do poprzedniego programu narzędziowego. Zwracamy przy tym uwagę, że program dzisiejszy odwzorowuje tylko pierwszą połowę możliwości dźwiękowych C 16 i Plus 4.

Dla lepszego poznania swojego komputera, na podstawie prezentowanego programu, oraz artykułu (patrz tabela) opublikowanego w naszym pierwszym numerze (listopad 1990r), możecie pokusić się o napisanie programu dla drugiej połowy skali dźwięków, lub uzupełnić odpowiednio nasz program. Sądzimy, że będzie to dobre ćwiczenie. Program wpisujemy do komputera i nagrywamy go na taśmę, lub dyskietkę znanymi komendami.

- SAVE "GRAJACY COMMODORE", 1 - taśma
- SAVE "GRAJACY COMMODORE", 8 - dyskietka

Dla lepszej czytelności programu wprowadziliśmy następujące oznaczenia;

- góra, dół, lewo, prawo - należy wcisnąć odpowiedni klawisz kursora po uprzednim wpisaniu cudzysłowu.
- SPC - spacja, 2SPC - dwie spacje itd.
- SU,S*,S-, - przy wciśniętym klawiszu SHIFT należy wcisnąć klawisz z odpowiednią literą, lub znakiem semigraficznym, 2SU - dwukrotnie należy wcisnąć odpowiednią literę itd.
- CC,CA itd. - jak wyżej, lecz zamiast klawisza SHIFT, wciskamy klawisz COMMODORE.
- nie piszemy przecinków występujących w tekście tytułu.

Program zajmuje 16243 bity i jest przeznaczony dla C 116 i C 16 z pamięcią RAM 64 KB.

```
10   SCNCLR:GOSUB 1590:REM - TYTUL -
20   GRAPHIC 1,1
30   REM
40   REM - PLANSZA -
50   REM
60   BOX 1,0,0,319,199
70   CHAR 1,9,1,"GRAJACY (2SPC) COMMODORE
     C=16"
80   CHAR 1,13,2" "
90   CHAR 1,15,3,"DZWIEK (2SPC) CZAS"
100  CHAR 1,14,5, "SU,7S*,CR,4S*,SJ,",1
110  CHAR 1,14.6,"S-,7SPC,S-,4SPC,S-",1
120  CHAR 1,14,7,"S-,7SPC,S-,4SPC,S-",1
130  CHAR 1,14,8,"SJ,7S*,CE,4S*,SK",1
140  REM
150  REM - KLAWIATURA -
160  REM
170  DRAW1,12,80TO316,80:DRAW 1,12,13OTO316,130
180  FORI=12TO316STEP16:DRAW1,I,81TOI,130:
     NEXTI
190  BOX1,23,81,33,106,,1:BOX1,55,81,65,106,,1:BOX
     1,71,81,81,106,,1
200  FORI=113TO146STEP16:BOX1,I,81,(I8)2,106,,1:
     NEXTI
210  FORI=177TO193STEP16:BOX1,I,81,(I-8)-2,106,,1:
     NEXTI
220  FORI=225TO257STEP16:BOX1,I,81,(I-8)-2,106,,1:
     NEXTI
230  FORI=289TO308STEP16:BOX1,I,81,(I-8)-2,106,,1:
     NEXTI
240  CHAR 1,03,11,"2"
250  CHAR 1,07,11,"4"
260  CHAR 1,09,11,"5"
270  CHAR 1,13,11,"7"
280  CHAR 1,15,11,"8"
290  CHAR 1,17,11,"9"
300  CHAR 1,21,11,"A"
310  CHAR 1,23,11,"S"
320  CHAR 1,27,11,"F"
330  CHAR 1,29,11,"G"
340  CHAR 1,31,11,"H"
350  CHAR 1,35,11,"K"
360  CHAR 1,37,11,"L"
370  CHAR 1,02,15,"Q"
380  CHAR 1,04,15,"W"
390  CHAR 1,06,15,"E"
400  CHAR 1,08,15,"R"
410  CHAR 1,10,15,"T"
420  CHAR 1,12,15,"Y"
430  CHAR 1,14,15,"U"
440  CHAR 1,16,15,"I"
450  CHAR 1,18,15,"O"
460  CHAR 1,20,15,"P"
470  CHAR 1,22,15,"Z"
480  CHAR 1,24,15,"X"
490  CHAR 1,26,15,"C"
500  CHAR 1,28,15,"V"
510  CHAR 1,30,15,"B"
520  CHAR 1,32,15,"N"
530  CHAR 1,34,15,"M"
540  CHAR 1,36,15,","
550  CHAR 1,36,15,"."
560  CHAR 1,36,14,"<170
570  CHAR1,38,14,">"
580  REM
590  REM - OPIS -
600  REM
610  CHAR 1,1,17,"CZAS TRWANIA DZWIEKU USTA
     LASZ KURSOREM"
620  CHAR 1,18,19,"<197 -",1
630  CHAR 1,1,21,"KLAWISZEM 'RETURN'- START
     GRANIA."
640  CHAR 1,1,22,"SPACJA-POWROT DO CZASU
     BRZMIENIA."
650  CHAR 1,1,23,"ESC - KONIEC."
660  REM
670  REM - USTALANIE CZASU BRZMIENIA -
680  REM
690  GETY$
700  T=PEEK (336)
710  CHAR1,23,6,STR$ (T)+"2SPC",,1
720  IFY$=CHR$ (29)AND PEEK(336)<THEN POKE
     336, PEEK(336)+1
```

```
730 IFY$=CHR$ (157) ANDPEEK (336)0THENPOKE
    336,PEEK (336)-1
740 IFY$=CHR$ (13) THEN 760
750 GOTO690
760 REM
770 REM - WARTOSCI NUT - GRANIE -
780 REM
790 GETKEYY$:VOL8
800 IFY$=CHR$ (27) THENGOSUB 1550:GOTO1560
810 IFY$="1" THENGOSUB 1550
820 IFY$="3" THENGOSUB 1550
830 IFY$="6" THENGOSUB 1550
840 IFY$="0" THENGOSUB 1550
850 IFY$="LEWO" THENGOSUB 1550
860 IFY$="PRAWO" THENGOSUB 1550
870 IFY$="GORA" THENGOSUB 1550
880 IFY$="DOL" THENGOSUB 1550
890 IFY$=CHR$ (20) THENGOSUB 1550
900 IFY$="e" THENGOSUB 1550
910 IFY$="+" THENGOSUB 1550
920 IFY$="-" THENGOSUB 1550
930 IFY$="DOM" THENGOSUB 1550
940 IFY$="D" THENGOSUB 1550
950 IFY$="J" THENGOSUB 1550
960 IFY$=":" THENGOSUB 1550
970 IFY$=":" THENGOSUB 1550
980 IFY$="*" THENGOSUB 1550
990 IFY$=CHR$ (13) THENGOSUB 1550
1000 IFY$="/" THENGOSUB 1550
1010 IFY$="&" THENGOSUB 1550
1020 IFY$="=" THENGOSUB 1550
1030 IFY$="Q" THENC=7:GOSUB 1380
1040 IFY$="2" THENC=64:GOSUB 1380
1050 IFY$="W" THENC=118:GOSUB 1380
1060 IFY$="E" THENC=169:GOSUB 1380
1070 IFY$="4" THENC=217:GOSUB 1380
1080 IFY$="R" THENC=262:GOSUB 1380
1090 IFY$="5" THENC=305:GOSUB 1380
1100 IFY$="T" THENC=345:GOSUB 1380
1110 IFY$="Y" THENC=383:GOSUB 1380
1120 IFY$="7" THENC=419:GOSUB 1380
1130 IFY$="U" THENC=453:GOSUB 1380
1140 IFY$="8" THENC=485:GOSUB 1380
1150 IFY$="I" THENC=516:GOSUB 1380
1160 IFY$="9" THENC=544:GOSUB 1380
1170 IFY$="O" THENC=571:GOSUB 1380
1180 IFY$="P" THENC=596:GOSUB 1380
1190 IFY$="A" THENC=620:GOSUB 1380
1200 IFY$="Z" THENC=643:GOSUB 1380
1210 IFY$="S" THENC=664:GOSUB 1380
1220 IFY$="X" THENC=685:GOSUB 1380
1230 IFY$="C" THENC=704:GOSUB 1380
1240 IFY$="F" THENC=722:GOSUB 1380
1250 IFY$="V" THENC=739:GOSUB 1380
1260 IFY$="G" THENC=755:GOSUB 1380
1270 IFY$="B" THENC=770:GOSUB 1380
1280 IFY$="H" THENC=784:GOSUB 1380
1290 IFY$="N" THENC=798:GOSUB 1380
1300 IFY$="M" THENC=810:GOSUB 1380
1310 IFY$="K" THENC=822:GOSUB 1380
1320 IFY$="," THENC=834:GOSUB 1380
1330 IFY$="L" THENC=844:GOSUB 1380
1340 IFY$="." THENC=854:GOSUB 1380
1350 IFY$=" " THEN690
1360 SOUND1,C,T
1370 GOTO790
1380 REM
1390 REM - NUTY I CZAS - PISANIE -
1400 REM
1410 IFC=169ORC=596ORC=810THENZ$="(2SPC)C
     (4SPC)"
1420 IFC=217ORC=620ORC=822THENZ$="CIS,DES"
1430 IFC=262ORC=643ORC=834THENZ$="(2SPC)D
     (4SPC)"
1440 IFC=305ORC=664ORC=844THENZ$="DIS,ES"
1450 IFC=345ORC=685ORC=854THENZ$="(2SPC)E
     (4SPC)"
1460 IFC=383ORC=704THENZ$="(2SPC)F(4SPC)"
1470 IFC=419ORC=722THENZ$="FIS,GES"
1480 IFC=453ORC=739THENZ$="(2SPC)G(4SPC)"
1490 IFC=485ORC=755THENZ$="GIS,AS"
1500 IFC=7ORC=516ORC=770THENZ$="(2SPC)A
     (4SPC)"
1510 IFC=64ORC=544ORC=784THENZ$="AIS,B
     (2SPC)"
1520 IFC=118ORC=571ORC=798THENZ$="(2SPC)H
     (4SPC)"1530 CHAR1,16,6,STR$(C)+"2SPC",1
1540 CHAR1,15,7,Z$,1:RETURN
1550 VOL0:C=0:CHAR1,16,6,"2SPC",1:CHAR1,15,7,"7
     SPC",1:RETURN
1560 GRAPHIC0:SCNCLR:PRINT"1ODOL,22SPC,,"
1570 PRINT"10PRAWO,DZIEKUJE - CZESC(2SPC)III"
1580 PRINT"13PRAWO'3SPC'":END
1590 PRINT"5DOL":A$="5PRAWO"
1600 PRITA$,,SU,S*,SI,CA,S*,SI,SU,S*,SI,SU,S*,CS,
     SU,S*,SI,SU,S*,SI,CR,SPC,CR"
1610 PRINTA$"S-,2SPC,S-,SPC,S-,S-,SPC,S-,2SPC,S-
     ,S-,SPC,S-,S-,2SPC,S-,SPC,S-"
1620 PRITA$"S,SPC,CS,CQ,S*,SK,CQ,S*,CW,2SPC,S-
     ,CQ,S*,CW,S-,2SPC,SJ,CR,SK"
1630 PRINTA$"S-,SPC,S-,S-,,SPC,S-,SPC,S-,2SPC,S-
     ,S-,SPC,S-,S-,3SPC,S-"
1640 PRINTA$"SJ,S*,SK,CE,SPC,,CE,SPC,CE,2SPC,
     S-,CE,SPC,,SJ,S*,SK,SPC,CE,SPC"
1650 PRINTA$"11SPC,S-"
1660 PRINTA$"10SPC,SJ,SK"
1670 PRITA$"SU,S*,SI,SU,S*,SI,SU,SI,SU,SI,SU,SI,SU,
     SI,SU,S*,SI,CA,S*,SI,SU,S*,SI,CA,S*,SI,CA,S*,CS"
1680 PRINTA$"S-,2SPC,S-,SPC,10S-,SPC,2S-,SPC,2S-
     ,SPC,2S-,SPC,2S-,2SPC"
1690 PRINTA$"S-,2SPC,S,SPC,2S,SJ,SK,2S,SJ,SK,2S-
     ,SPC,2S-,SPC,2S-,SPC,2S-,CQ,S*,SK,CQ,CW,SPC"
1700 PRINTA$"S-,2SPC,S-,SPC,2S,2SPC,2S,2SPC,2S-
     ,SPC,2S-,SPC,2S-,SPC,2S-,,SPC,S-,2SPC,CD,CF"
1710 PRITA$"SJ,S*,SK,SJ,S*,SK,CE,2SPC,2CE,2SPC,
     CE,SJ,S*,SK,CZ,S*,SK,SJ,S*,SK,CE,SPC,,CZ,S*,
     CX,CC,CV"
1720 FORT=1TO5000:NEXTT:RETURN
```

Jan Siedlecki

SŁÓW PARĘ O TYM, JAK GRA TWÓJ KOMODOREK

Idę o zakład, że bardzo niewielu użytkowników Commodore 64 zastanawia się nad tym, jak to właściwie jest, że ich poczciwy komputer potrafi tak ładnie grać.

Jeżeli posiadacie lub macie dostęp do starych programów (np. gry z roku 1985/86) to zwróćcie uwagę nie na melodię, a na używane dźwięki. Wraz z ewoluowaniem procedur grających znacznie poprawiała się jakość dźwięku.

Układ dźwiękowy, który zamontowano w C-64 jest prostym syntezatorem, który bez specjalnej pomocy nie potrafi z siebie wydobyć bardzo skomplikowanych brzmień. „Wobec tego, skąd biorą się dźwięki perkusji, basu itp.?" - możecie spytać. Odpowiedź jest prosta, wszystko to zależy od jakości procedury grającej.

Ogólnie można założyć, że w momencie wywołania samego „grajka", procesor wykonuje skok do czterech podprogramów - 3 procedur obsługujących zapis linii melodycznej (w dużym uproszczeniu) i procedury, która zajmuje się syntezą dźwięku. W przypadku wymienionych 3 procedur „odczytu muzyki" ich opis można ograniczyć do stwierdzenia, że każda odpowiada jednemu generatorowi, zajmuje się odczytem kolejnej nuty, ustaleniem jej długości trwania i wysokości dźwięku, oraz pobraniem danych dotyczących sterowania syntezą dźwięku. Z reguły procedury obsługujące pracę generatorów pracują i przetwarzają dane niezależnie od siebie.

Jednak z punktu widzenia odbiorcy, ważne jest to co słyszy, a nie to, co się tam w środku komputera „mieli" i do tego służy czwarty podprogram odpowiedzialny za syntezę. Jego główne zadanie polega na tym, by nie tylko wpisywać do SID-a dane dotyczące wysokości poszczególnych dźwięków, kształtu fali itp., ale również modyfikować parametry podczas trwania dźwięku, co pozwala na „wyciśnięcie" z SID-a o wiele więcej, niż to wynika z jego opisanych możliwości. Jest to bardzo ważna cecha procedur muzycznych napisanych w kodzie maszynowym, żadne procedury grające napisane w języku BASIC nigdy nie sprostają wymogom prędkości.

Jak więc można wykorzystać potencjalne możliwości drzemiące w zastosowaniu resyntezy przy okazji każdego wywołania muzyki? Najprościej przedstawić to można na przykładzie makroinstrukcji zastosowanych w edytorze muzycznym VOICETRACKER.

Zastanówmy się, jak syntetyzuje się np. dźwięk perkusji.Aby tego dokonać należy, operować kilkoma kształtami fal dźwiękowych (zmienianych co każde wywołanie „grajka"), co schematycznie może wyglądać np. w ten sposób:

1. zaczynamy szumem ($80)
2. fala prostokątna ($40)
3. fala prostokątna ($40)
4. i kończymy ponownie szumem ($40)

Jest to oczywiście schemat, a w przypadku VOICETRACKER'a rozwiązano to w ten sposób, że każdemu opisowi kształtu fali towarzyszy odpowiadający jej parametr wysokości dźwięku, oraz parametr sterowania filtrami. W zapisie wygląda to np. w ten sposób:

```
00: 81 c5 00
01: 41 a8 00
02: 41 a4 00
03: fe 00 00
```

...co należy zinterpretować następująco:

00. dźwięk: szum; wysokość: c5; filtr: 00 (nic)
01. dźwięk: prostokąt; wysokość: a8; filtr: 00
02. dźwięk: prostokąt; wysokość: a4; filtr: 00
03. wartość $fe-czyli zakończenie makroinstrukcji (ostatnie parametry są wykonywane aż do wygaśnięcia dźwięku).

Makroinstrukcje mają to do siebie, że poszukiwane dźwięki uzyskuje się poprzez dość żmudne próbowanie różnych wartości parametrów, co po jakimś czasie zaowocuje dobrym, oryginalnym brzmieniem. W tym konkretnym przypadku, makroinstrukcja została zakończona wartością $fe w miejscu, które zwykle oznacza kształt fali.

Warto wiedzieć, że gdy umieścimy w tym miejscu wartość $ff, to spowoduje to wykonywanie tej makroinstrukcji ponownie od pozycji oznaczonej zerem (czyli od początku). Jest to bardzo przydatne np. w przypadku symulowania trójdźwięków przy pomocy jednego generatora, co odbywa się np. w następujący sposób:

```
00: 41 00 00
01: 41 07 00
02: 41 0c 00
03: ff 00 00
```

Drugi parametr, który oznacza wysokość dźwięku działa w ten sposób, że o ile jego wartość jest mniejsza od $80, to oznacza ilość dodanych półtonów do wysokości dźwięku, określonej przez zapis w patternie, a w sytuacji, gdy wartość jest większa lub równa $80, to grany jest dźwięk w/g wzoru:

grany dźwięk=C-0 (najniższy dźwięk)+(parametr-$80)

Rozwiązanie takie, pozwala utrzymywać stałe wysokości dźwięków, pomimo ewentualnych transpozycji całych patternów w zapisie dla któregoś z generatorów, co jest bardzo ważne np. przy imitowaniu dźwięku perkusji itp.

Paweł Sołtysiński

JAK ZROBIĆ DEMO (6)

Skoro już jesteśmy w posiadaniu prawidłowo zdefiniowanego kompletu znaków (patrz poprzedni odcinek naszego cyklu), możemy przystąpić do napisania procedury samego scroll'a, który będzie przewijał tekst w dwóch górnych liniach ekranu.

Na początek założenia: procedura ma umożliwiać zmianę prędkości przewijania tekstu oraz opcję pauzy (tzn. chwilowego zatrzymania przewijania). Poniżej podano, jak zwykle tekst źródłowy programu napisanego w Turbo Assemblerze.

```
    *= $6000            ;przykładowy adres startu

SCROLL      =$D016
RASTER      =$D012
HIRASTER    =$D011
RASTERSET   =$D01A
RASTCONF    =$D019
CHARBANK    =$D018
IRQEXIT     =$EA31
VECT        =$FB    ;młodszy bajt wektora dla tekstu
SCRL        =$FD

    ;IRQ - start (ustawienie parametrów)
            SEI
            LDA #$7F
            STA $DC0D
            LDX #$00
            STX $DC0E
            INX
            STX RASTERSET
            LDA #$1B
            STA HIRASTER
            LDA #$32
            STA RASTER
            LDA #$18            ;ustawienie nowego ge-
                                ;neratora znaków
            STA CHARBANK        ;pod adresem $2000
            LDA #<IRQ

            LDX #IRQ
            STA $0314
            STX $0315
            JSR TEXTSET
            CLI
            RTS

    ;IRQ - procedura obsługi przerwań
IRQ         LDA SCRL
            STA SCROLL
            LDA #$42
LOOP1       CMP RASTER
            BNE LOOP1
            LDA #$C8
            STA SCROLL
            JSR SCRDRIVER
            INC RASTCONF
            JMP IRQEXIT
    ;SCROLL - procedura właściwa
SCRDRIVER   LDA #$00            ;test pauzy (zero-nie ma)
            BEQ LOOP2
            DEC SCRDRIVER+1     ;jest pauza-zmniejsz
                                ;o jeden
            RTS
LOOP2       LDA SCRL
            SEC
SPD         SBC #$01            ;01-prędkość pierwsza
            STA SCRL
            AND #$08
            BNE LOOP3           ;przesunięto cały znak?
            RTS                 ;RTS, jeśli nie przesunięto
                                ;znaku
LOOP3       LDA SCRL
            CLC
            ADC #$08
            STA SCRL
            LDX #$00            ;przesunięcie dwóch linii
                                ;ekranowych
LOOP4       LDA $0401,X         ;o cały znak w lewą stro-
                                ;nę
            STA $0400,X
            INX
            CPX #$4F            ;przesunięto wszystkie
                                ;linie?
            BNE LOOP4           ;jeśli nie, to idź do LOOP4
HALF        LDA #$00            ;która połowa dużego
                                ;znaku?
            BEQ LOOP5           ;zero - pierwsza połowa
                                ;nowego znaku
ZNAK        LDA #$00
            CLC
            ADC #$40
HOPS        STA $0427           ;wstaw znak do pierwszej
                                ;linii
            CLC
            ADC #$80
            STA $044F           ;wstaw znak do drugiej linii
            DEC HALF+1
            RTS
LOOP5       LDY #$00
            LDA (VECT),Y        ;odczytanie kolejnej litery
                                ;tekstu
            BNE LOOP6           ;idź do LOOP6, jeśli nie
                                ;koniec
            JSR TEXTSET1
            JMP LOOP5
LOOP6       CMP #$50            ;jeśli litera „P", to pauza
            BNE LOOP7           ;jeśli nie, to idź do LOOP7
            LDA #$80
            STA SCRDRIVER+1     ;ustaw pauzę na $80
            JSR VECTINCR        ;zwiększ wektor tekstu
                                ;o jeden
            RTS
LOOP7       CMP #$40            ;czy to zmiana prędkości?
            BCC LOOP8           ;jeśli nie, to idź do LOOP6
            AND #$07            ;odczytanie prędkości
            STA SPD+1           ;i wstawienie jej do
                                ;programu
            JSR VECTINCR        ;zwiększ wektor tekstu
                                ;o jeden
            JMP LOOP5           ;i pobierz następny znak
```

```
LOOP8      LDX #$02
           STX HALF+1          ;ustaw znacznik połowy
           JSR VECTINCR        ;zwiększ wektor tekstu
                               ;o jeden
           STA ZNAK+1          ;wstaw aktualny znak do
                               ;programu
           JMP HOPS            ;i wydrukuj jego 1 połowę
VECTINCR   INC VECT
           BNE LOOP9
           INC VECT+1
LOOP9      RTS
TEXTSET    LDA #$C7            ;wstaw wartość
                               ;standardową do
           STA SCRL            ;wskaźnika przesuwania
           LDX #$01
           STX SPD             ;ustaw 1 prędkość
           DEX
           STX SCRDRIVER+      ;wyzeruj pauzę
           STX HALF+1          ;i znacznik połowy
TEXTSET1   LDA #<TEXT          ;ustaw wektor na
                               ;początek tekstu
           STA VECT
           LDA #TEXT
           STA VECT+1
           RTS
TEXT
```

Pod etykietą TEXT należy umieścić tekst dla naszego scroll'a, pamiętając o kilku zasadach:

- tekst MUSI być napisany w kodach ekranowych (najłatwiej napisać tekst na ekranie, a potem wstawić go w odpowiednie miejsce za pomocą instrukcji TRANSFER dowolnego monitora pamięci)

- kod 0 (zero) oznacza koniec tekstu (procedura startuje wtedy od początku)

- znak o kodzie $50 (pisząc na ekranie-litera „P" z Shift'em) jest znacznikiem pauzy. Po jego napotkaniu procedura zawiesza scrolling na okres 128 wywołań procedury, po czym wznawia przewijanie tekstu.

- znaki o kodach od $41 do $47 (od „A" do „G" z Shift'em) są z kolei znacznikami prędkości przewijania tekstu. Odpowiednio ich używajac można uzyskać ciekawe efekty.

Myślę, że samo prześledzenie napisanej powyżej procedury będzie interesującymym zajęciem dla początkującego programisty, ze względu na dość ciekawą technikę automodyfikacji procedury, która wstawia odczytane i zinterpretowane dane prosto... w samą siebie, powodując zmiany w pracy programu. Powodzenia!



Commodore 64 nie posiada w spisie instrukcji BASIC żadnych rozkazów pozwalających na wygodne korzystanie z dostępnych trybów graficznych. Potencjalny użytkownik musi więc zdać się na jedno z dostępnych mu rozszerzeń interpretera (np.Simons Basic), lub napisać własne procedury, które pozwolą na „okiełznanie" nieswornego komputera. Jak zapewne wiadomo, w trybie wielobarwnym (tzw. multicolor) mamy możliwość korzystania ze wszystkich 16 kolorów. Nie znaczy to, że możemy postawić obok siebie kilkanaście punktów, każdy w innym kolorze. Dlaczego? Wyjaśnienie jest proste - do zapisu informacji o znajdujących się w obrębie danego znaku kolorach (oprócz koloru tła) używa się dwóch bajtów, zwanych dalej atrybutami. Dla przypomnienia warto przejrzeć jeden z wcześniejszych numerów „C64+4", gdzie w tym samym dziale wyjaśniałem sposoby włączania trybów graficznych na przykładzie procedury pokazującej obrazki wykonane przy pomocy programu Advanced Art Studio.

Jak pamiętacie, do prawidłowego opisania komputerowi wyglądu obrazka, oprócz tzw. bit-map'u potrzebne były jeszcze dwa zbiory danych, z których wielkość każdego odpowiadała rozmiarom standardowego ekranu tekstowego (a więc $03E8).
Jeden z tych zbiorów umieszczany był w polu atrybutów ($D800 - $DBE8), a drugi w obszarze aktualnie ustawionego ekranu tekstowego. Takie ustawienie powoduje przyznanie każdemu znakowi na ekranie graficznym (określenie „znak" nie jest precyzyjne - oznacza ono obszar 8 - miu bajtów, które będąc wyświetlanymi jeden pod drugim, dają nam obszar równy wielkością polu przeznaczonemu normalnie na dowolny znak w trybie tekstowym) dwóch wspomnianych wyżej atrybutów (po jednym bajcie z pola ekranu i pola atrybutów), co daje możliwość opisania 3 kolorów.

Pomyślcie sobie teraz, jak to byłoby fajnie, gdyby każdemu z tych bit - map'owych 8 - miu bajtów (tworzących „znak") przypisać własny ekran atrybutów! Oznaczałoby to, że kolory każdego znaku opisywane by były przez 9 bajtów (jeden w polu atrybutów, a pozostałe osiem na ośmiu odzielnych ekranach tekstowych)! Teoretycznie sprawa jest prosta - wystarczy napisać procedurę, która np. w obsłudze przerwań graficznych będzie przełączać położenie ekranów tekstowych co każdą linię ekranu. Łatwiej jednak powiedzieć niż zrobić, jak się okazuje, układ graficzny w C - 64 w normalnych warunkach pozwala na przełączanie ekranu tylko co osiem (!) linii, co w praktyce oznacza, że wszystkie próby zmian dokonywane częściej, niż co 8 linii (oczywiście z punktu widzenia strumienia elektronów tworzących obraz na ekranie) nie dają zamierzongo efektu, co najwyżej po minięciu kolejnych 8 linii położenie ekranu jest ustalane na podstawie ostatnio wpisanej tam wartości. Cała więc trudność polegała na napisaniu procedury (a w zasadzie na znalezieniu techniki), która pozwalała by na „zmuszenie" VIC - a do zmiany położenia ekranu co 1 linię. Zamierzenie prawie się udało - napisałem „prawie", bo są także skutki uboczne: 3 pierwsze pionowe linie znakowe od lewej strony stają się niemożliwe do wykorzystania, ale „coś za coś"...

A teraz kilka uwag do umieszczonej poniżej procedury FLI (to skrót od angielskiej nazwy tej techniki: Flexible Line Interpretation):

- procedura została napisana przy użyciu Turbo Assemblera, ale bez przeszkód może być wpisana pod dowolny inny assembler, co najwyżej z drobnymi zmianami;

- dane do FLI - obrazka zajmują CAŁY bank VIC - a od adresu $4000 do $7FFF i ogólnie wygląda to tak:

od $4000 do $5FFF - 8 ekranów atrybutów (ekrany tekstowe)

od $6000 do $7FFF - bit - map

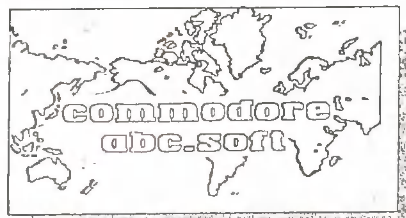
- dane dla pola atrybutów (pod $D800) należy wpisać samemu, bo ta procedura to tylko „wyświetlacz", a nie jakiś specjalizowany „show - pic".

```
*= $1000
```

```
         LDA     #$06
         STA     $02
         LDA     #$00
         STA     $03
         LDX     #$00
LOOP1    LDA     #$38
         CLC
         ADC     $02
         STA     $1100,X      ;ustawianie tablicy dla
                              ;$D011
         LDA     #$0E
         ORA     $03
         STA     $1200,X      ;ustawianie tablicy dla
                              ;$D018
         LDA     $02
         CLC
         ADC     #$01
         AND     #$07
         STA     $02
         LDA     $03
         CLC
         ADC     #$10
         AND     #$70
         STA     $03
         INX
         BNE     LOOP1
         SEI
         LDA     #$7F
         STA     $DC0D
         LDX     #$00
         STX     $DC0E
         INX
         STX     $D01A
         LDA     #$3B
         STA     $D011
         LDA     #$30
         STA     $D012
         LDA     #$D8
         STA     $D016
         LDA     #$96
         STA     $DD00
         LDA     #<IRQ
         STA     $0314
         LDA     #IRQ
         STA     $0315
LOOP2    BIT     $D011
         BPL     LOOP2
LOOP3    BIT     $D011
         BMI     LOOP3
         CLI
LOOP4    JMP     LOOP4
IRQ      LDX     #$00         ;początek pętli FLI
LOOP5    LDA     $1100,X
         LDY     $1200,X
         STA     $D011
         STY     $D018
         INX
         CPX     #$C0
         BNE     LOOP5        ;koniec pętli FLI
         INC     $D019
         JMP     $EA7E
```

Jak daje się łatwo zauważyć, pominąłem w tym artykule zupełnie sprawę rysowania w tym trybie. Na rynku istnieje jednak co najmniej kilka programów, które to ułatwiają (np. FLI - GRAPH grupy Black Mail z Holandii).

Paweł (Polonus) Sołtysiński



UWAGA UŻYTKOWNICY PROGRAMU VOICETRACKER V 4.0!

Niektórzy z Was sygnalizują, że nie wiedzą jak uruchomić demonstracje muzyczne znajdujące się na dyskietce lub taśmie. Otóż demonstracje te należy wgrywać nie jako samodzielne programy, ale jako moduły programu. Po wgraniu Voicetracker'a i wejściu do głownego okna edycji wybieramy opcję taśma lub dysk, a następnie wprowadzamy komendę LOAD (klawisz „l"). Komputer zapyta o tytuł pliku jaki chcemy wprowadzić (wpisujemy np. TUNE 01), a następnie wczyta go do pamięci. Po przejściu do głównego menu klawiszem F1 uruchamiamy daną demonstrację.

REDAKCJA

Kosmos w jednym duszku!

Przypuszczam, że każdy lubi popatrzeć na niebo pełne gwiazd (zakładając dobrą widoczność). Ładne gwiazdki zobaczyć można też w niektórych programach demonstracyjnych, najczęściej u znajomego, który ma Amigę - czasami aż przykro patrzeć, jak niedoświadczonego użytkownika Commodore 64 zazdrość zjada. Należy jednak pamiętać, że w przypadku programów demonstracyjnych część prezentowanych efektów nie jest bezpośrednio zależna od możliwości sprzętu, a jedynie od pomysłowości programisty.

Tak właśnie ewoluował pomysł „przestrzennie" przesuwających się punktów (gwiazd) w realizacji dla C - 64. Na początku problemem było stworzenie więcej niż 8 gwiazdek (tyle tylko jest sprite'ów...), potem nauczono się powielać ich liczbę przez wymuszenie wyświetlania duszków w kilku miejscach ekranu podczas trwania jednej tzw. ramki. Ostatnim etapem gwiazdek na duszkach są gwiazdy na... jednym duszku! Tak, tak - to możliwe! Wystarczy wpisać zamieszczony poniżej program i sami zobaczycie.

Procedurę można później wykorzystać np. przy pisaniu własnych programów w BASIC'u.

Bardziej zainteresowanym polecam wnikliwe obejrzenie budowy programu, a wszystko stanie się (mam nadzieję) jasne.

A oto tekst spod Turbo Assemblera:

```
                *= $C000
LDX             #$3F
LDA             #$00
LOOP1           STA     $0340,X
                DEX
                BPL     LOOP1
                INX
LOOP2           LDA     $E000,X
                AND     #$01
                BNE     LOOP10
                LDA     #$80
                .BYTE   $2C
LOOP10          LDA     #$C0
                STA     $0340,X
                TXA
                CLC
                ADC     #$06
                TAX
                CPX     #$3F
                BCC     LOOP2
                LDX     #$00
LOOP7           LDA     $F400,X
                AND     #$03
                CLC
                ADC     #$01
                STA     TAB0,X
                LDA     $E200,X
                PHA
                AND     #$01
                STA     TAB2,0
                BNE     LOOP8
                PLA
                JMP     LOOP9
LOOP8           PLA
                AND     #$7F
LOOP9           STA     TAB1,X
                INX
                CPX     #$CD
                BCC     LOOP7
                SEI
                LDA     #$7F
                STA     $DC0D
                LDX     #$00
                STX     $DC0E
                INX
                STX     $D01A
                STX     $D015
                STX     $D027
                STX     $D01B
                DEX
                STX     $D01C
                STX     $D01D
                STX     $D017
                LDA     #$1B
                STA     $D011
                LDA     #$30
                STA     $D01
                LDA     #<IRQ
                STA     $0314
                LDA     #IRQ
                STA     $0315
                LDA     #$0D
                STA     $07F8
                CLI
                RTS
IRQ             LDY     #$30
LOOP4           INY
                INY
                LDA     TAB1 - $32,Y
                TAX
                LDA     TAB2 - $32,Y
LOOP3           CPY     $D012
                BNE     LOOP3
                STY     $D001
                STA     $D010
                STX     $D000
                CPY     #$FE
                BCC     LOOP4
                LDX     #$00
LOOP5           LDA     TAB1,X
                CLC
                ADC     TAB0,X
                STA     TAB1,X
                BCC     LOOP11
                LDA     #$01
                STA     TAB2.X
                BNE     LOOP6
LOOP11          LDY     TAB2,X
                BNE     LOOP6
                CMP     #$79
                BCC     LOOP6
                LDA     #$00
                STA     TAB1,X
                STA     TAB2,X
LOOP6           INX
                INX
                CPX     #$CD
                BCC     LOOP5
                NC      $D019
                JMP     $EA31
TAB0
TAB1            =TAB0+$CD
TAB2            =TAB1+$CD
```

Paweł Sołtysiński

Podręczny DOS

Artykuł ten i zamieszczony w nim program powinny zainteresować nie tylko posiadaczy stacji dysków. Ze względu na pewne rozwiązanie programowe może on być interesujący dla wszystkich początkujących programistów.

Przejdźmy jednak do omówienia „dyskowej" części zagadnienia. Od razu muszę nadmienić, że problem nie dotyczy posiadaczy kart typu FINAL, lub ACTION REPLAY, ponieważ obsługa komend dla stacji dysków została tam już wygodnie zrealizowana. Przy założeniu, iż użytkownik z różnych względów takowej karty nie posiada, wszelkie operacje ze stacją z poziomu BASIC'a stają się dość uciążliwe, bo ileż można pisać np.: OPEN 1,8,15, „S: nazwa1,nazwa2,...": close 1 nie mówiąc już o katalogu dowolnej dyskietki, który za pomocą LOAD „$", i za każdym razem musimy umieszczać w pamięci operacyjnej, jako program w języku BASIC (tzn. dane są w tym właśnie obszarze umieszczane), co po prostu niszczy nasz program, o ile się tam wówczas znajdował.

Aby temu zaradzić, postanowiłem napisać taki program, który pozwolił by mi na obsługę DOS-u (Disk Operating System) wolną od wymienionych powyżej wad. Ku mojemu zaskoczeniu, całość programu wraz z obsługą dodatkowej komendy BASIC okazała się bardzo krótka (sama procedura zajmuje tylko 272 bajty). Prezentowana tu wersja po wpisaniu posiada pierwszą linię w języku BASIC, która zawiera rozkaz uruchomienia programu w j. maszynowym, który uaktywnia właściwą procedurę. Całość należy wpisać do pamięci komputera przy użyciu dowolnego monitora pamięci, poczynając od adresu $0801. Po prawidłowym wpisaniu program należy nagrać na dyskietkę rozkazem:

S „disk - dos", 08, 0801, 0932

```
:0801   0b 08 c7 07 9e 32 30 36
:0809   31 00 00 00 a2 00 bd 21
:0811   08 9d 80 ce bd 31 08 9d
:0819   90 ce e8 d0 f1 4c 00 ce
:0821   a2 01 8e 86 02 ca 8e 20
:0829   d0 8e 21 d0 a9 08 85 ba
:0831   a9 ce 8d 19 03 a9 bd 8d
:0839   18 03 a9 4c a2 ca a0 ce
:0841   85 7c 86 7d 84 7e a9 ad
:0849   a0 ce 4c 1e ab 0d 44 49
:0851   53 4b 2d 44 4f 53 2f 36
:0859   34 2b 34 0d 00 58 20 1b
:0861   a5 a2 f1 8e 86 02 9a 4c
:0869   74 a4 c9 40 f0 0a c9 3a
:0871   b0 03 4c 80 00 4c 8a 00
:0879   20 73 00 a0 00 b1 7a f0
:0881   03 c8 d0 f9 c0 00 d0 21
:0889   a9 0d 20 d2 ff a5 ba 20
:0891   b4 ff a9 6f 20 96 ff 20
:0899   a5 ff 20 d2 ff c9 0d d0
:08a9   a4 a0 00 b1 7a c9 24 f0
:08b1   03 4c 6d cf a9 0d 20 d2
:08b9   ff a9 01 a2 60 a0 a3 20
:08c1   bd ff a2 08 a0 00 20 ba
:08c9   ff 20 c0 ff a2 01 20 c6
:08d1   ff 20 cf ff 20 cf ff 20
:08d9   cf ff 20 cf ff a5 90 d0
:08e1   21 20 cf ff aa 20 cf ff
:08e9   20 cd bd 20 3b ab 20 cf
:08f1   ff 20 d2 ff d0 f8 a9 0d
:08f9   20 d2 ff ad 01 dc c9 7f
:0901   d0 d5 a9 01 20 c3 ff 20
:0909   cc ff 4c 04 cf a5 ba 20
:0911   b1 ff a9 6f 20 93 ff a0
:0919   00 b1 7a f0 0e 20 a8 ff
:0921   c8 d0 f6 a0 00 b1 7a c9
:0929   55 f0 03 20 ae ff 4c 04
:0931   cf 00
```

Po uruchomieniu program instaluje nową komendę, rozpoznawalną przez edytor BASIC'a. Komenda ta składa się ze znaku „@" i występującego po nim argumentu, np.:

@$ - wyświetla na ekranie katalog dyskietki

@S: nazwa - kasuje z dysku program „nazwa"

@ - (bez argumentu) powoduje wyświetlenie raportu stacji.

W sumie rozkaz ten zastępuje operację OPEN/CLOSE z języka BASIC - treść rozkazu wysyłanego do stacji należy po prostu wpisać po znaku „@" i wcisnąć RETURN.

Na początku tego artykułu napisałem o tym, że początkujący programiści znajdą tutaj ciekawe rozwiązanie programowe. Aby zainstalować własny rozkaz w j. BASIC, należy zmienić dla własnych celów wektory systemu i napisać własne procedury obsługi nowych rozkazów. Najczęściej do tych celów zmieniany jest wektor $0308/$0309. W moim programie zdecydowałem się nie zmieniać żadnego z tych wektorów - moim celem stała się niewielka procedura umieszczona na pierwszej stronie pamięci od adresu $0073, służąca do odczytywania kolejnych bajtów programu. Jest ona przy uruchamianiu komputera przepisywana z pamięci ROM do RAM z wielu względów (m.in. ze względu na późniejsze adresowanie pamięci przy użyciu wektora zawartego w tej procedurze, np. LDA($7a),Y). Ciekawym proponuję prześledzenie działania tego programu, który w czasie działania znajduje się w obszarze od adresu $CE80 do $CF90.

Paweł Sołtysiński

Sztuczki i kruczki

W zachodnich magazynach „sztuczki i kruczki" (Tips und Tricks) mają swoje stałe miejsce. W działach tych opisywane są różne chwyty i pomysły, jak usprawnić działanie programów lub sprzętu. Okazuje się bowiem często, że prostymi metodami można poprawić ich właściwości. Dziś przedstawiamy kilka sztuczek w oparciu o nadesłane przez czytelników magazynu „64'er" materiały, a opublikowane w numerze 11/88 .

Poszerzony RAM - dysk

Użytkownicy rozszerzenia RAM'u 1700 (128 kB) szybko zauważą, że GEOS nie jest w stanie zastosować go jako RAM - dysku z powodu ograniczonej pojemności. Fakt ten skłania ich do poszukiwania większych rozszerzeń. Można jednakże do tego celu wykorzystać tę samą płytkę, wymieniając 16 układów pamięciowych na kości typu 41256 (120 ns).

Aby dokonać wymiany potrzebne jest doświadczenie przy lutowaniu, bowiem trzeba najpierw wylutować stare układy RAM (U2 do U17). Na ich miejsce wlutować podstawki o 16 nóżkach, a następnie wstawić w nie układy 41256 (120 ns). Na koniec należy usunąć połączenie J1, aby dokonane rozszerzenie mogło zostać wykryte.

Teraz już GEOS będzie w stanie wykorzystać nasze rozszerzenie jako RAM - dysk.

Dioda sygnalizująca błędy w 1581

Zapewne większości użytkowników 1581 przeszkadza fakt, że czerwona dioda sygnalizująca błędy w tym napędzie, w odróżnieniu od pozostałych napędów tylko bardzo słabo migocze. W zależności od oświetlenia ciężko jest czasami stwierdzić, czy błąd wystąpił czy też nie. Sprawdzamy, więc co jest przyczyną migotania, a nie włączania i wyłączania tej diody.

Gdy się dokładniej przyjrzymy schematowi stacji, szybko można stwierdzić, że dioda połączona jest przez opornik do układu wejścia/wyjścia. Drugi opornik przyłączony do napięcia +5V powoduje, że dioda jest ciągle włączona: czerwona dioda nie może więc zostać wyłączona.

Rozwiązanie problemu jest następujące: należy usunąć opornik R14 z płyty głównej. Dzięki brakującemu połączeniu z +5V dioda daje się teraz wyłączyć i jej migotanie można łatwo rozpoznać.

Błędy stacji 1571

Gdy moja stacja przestała poprawnie odczytywać dane z dyskietek (np. błąd odczytu nr 27) postanowiłem wykryć, co jest tego przyczyną. Nie mogło to być spowodowane błędami w programach, ponieważ dotychczas działały one bez zarzutu. Pozostaje więc sprzęt. Podejrzenie o zabrudzenie głowicy zapisująco - odczytującej okazało się nietrafne po jej przeczyszczeniu. Także kalibracja głowicy w serwisie nie przyniosła oczekiwanego rezultatu. W końcu pozostało tylko jedno źródło błędów: rozregulowanie odstępu głowicy od nośnika.

Sprężyna, które dociska głowicę zapisująco - odczytującą do powierzchni dyskietki może ulec wykrzywieniu, jeśli stacja pozostaje przez dłuższy czas z otwartym napędem, bądź gdy jest zamknięta bez włożonej dyskietki. A jeśli ta sprężyna ulegnie wykrzywieniu to zabraknie przeciwważącej siły dla głowicy z drugiej strony. W ten sposób mogą wystąpić zakłócenia zarówno przy zapisie, jak i odczycie.

W żadnym jednak wypadku nie należy próbować spowrotem odginać sprężyny, która dociska w dół górną głowicę, gdyż może to spowodować uszkodzenie zarówno głowicy, jak i dyskietki. Gdy stwierdzą taką usterkę najlepiej będzie oddać stację do serwisu. Aby uniknąć w przyszłości takich awarii należy się trzymać następującej zasady:

Stacja powinna mieć zawsze dyskietkę w środku oraz zamknięty klucz.

Pewniejsza współpraca z 1581

Zdarza się, że stacja 1581 uszkadza czasami pojedyncze ścieżki. Błąd ten występuje nieregularnie i w związku z tym jest trudny do wykrycia. Po dokładnej analizie uszkodzonych dyskietek udało się go jednak zlokalizować: stacja zapisuje poprawnie część ścieżki, a część w ogóle pomija. Z tego wynika, że procedura zapisująca zostaje w pewnym momencie przerwana. Owo przerwanie zapisu powoduje, że nie zgadza się suma kontrolna. Gdy próbuje się następnie odczytać uszkodzoną ścieżkę pojawiają się różnego rodzaju komunikaty o błędzie odczytu. Usterkę można usunąć przynajmniej częściowo, przy pomocy programu „Ratownik ścieżek" (listing na następnej stronie) - naprawia on pojedyncze ścieżki.

Jest to jednak niewielka pomoc, ponieważ ten program dopisuje tylko poprawną sumę kontrolną, natomiast nie przywraca utraconej informacji. Ścieżka staje się tylko ponownie zdolna do zapisu i odczytu.

Trzeba więc dotrzeć do źródła tych kłopotów. Po wnikliwej analizie listingu ROM'u stacji 1581 znaleziono w końcu przyczynę: w procedurze zapisu na dysk nie

```
0 INPUT "Uszkodzona sciezka i strona"; T,
  s:S=S*20+19
1 OPEN2, 8, 2, "#" : OPEN1, 8 15 : PRINT#1, "U1
  2 0"; T ; S
2 PRINT#1, "UO" + CHR$ (134) + CHR$ (2) +
  CHR$ (T-1) + CHR$ (10);
3 PRINT#1, CHE$ (T-1) + CHR$ (181) + CHR$
  (S) : CLOSE1
```

zostały zablokowane przerwania, przez co może ona zostać w każdej chwili zatrzymana. Przy usuwaniu tej usterki pojawił się nowy problem: test sumy kontrolnej podczas włączania stacji. Można go obejść poprzez przeskoczenie procedury testującej tą sumę. Wystarczy zmienić części ROM'u, tak by nie spowodować zawieszenia programu przy wykonywaniu tej procedury. Teraz można już wprowadzić rozkaz blokujący przerwania. Ponieważ jednak pamięci ROM nie można ponownie zapisać potrzebna będzie do zamiany pamięć EPROM (27256), oraz programator EPROM'u.

Trzeba zmienić 8 bajtów, które wyszczególniono w tabeli.

Adres	Stara wartość	Nowa wartość	Opis
$af8c	$3a	$00	Przeskoczenie testu sumy kontrolnej
$c160	$6c	$4c	
$c181	$48	$af	Skok do rozszerzenia
$c162	$00	$c5	
$c5af	$ff	$78	Blokada przerwań
$c5b0	$ff	$6c	
$c5b1	$ff	$48	Skok do wektora JRQ
$c5b2	$ff	$00	

W ROM'ie stacji 1581 pozostaje jeszcze jeden błąd objawiający się tym, że czasami nie następuje zapis sektora. Nie został on jeszcze wykryty. Jeśli ktoś z Was jest w stanie go zlokalizować, prosimy o kontakt.

Jeśli posiadacie własne „sztuczki i kruczki" piszcie do nas. Chętnie je opublikujemy.

Redakcja

PUBLIC DOMAIN PACK - TAPE NR1

* SINUSDATA - EDITOR
* FAST CRUNCHER V3
* ANAL S.C. IBEYOND
* VECTOR - VICTORY
* PUZZLENOID +4
* TUNE OF MONTH #1
* NIM
* STRZAŁKA 64+
* LOGO - WRITER V.2.0
* CAN'T TOUCH IKU!
* INTRO PRV
* BONZIEED !!
* ZAX PACKIS
* READ THIS FIRST
* COMMERCIAL BREAK
* 290 SPRITES!
* NOTE - ABOUT
* BAD NEWS NR2
* TO BAD NEWS...
* CONTACT CORNER!
* PROJEKT DUSZKÓW
* SYMPHONY NR 14
* SYMPHONY NR 15
* SYMPHONY NR 16
* SYMPHONY NR 17
* SYMPHONY NR 18
* SYMPHONY NR 19
* CRUISER/GIANTS
* NOTE > ANO < PADUA
* LET'S DYSP!
* FINALTAPE
* MUSIC - SEARCHER

W związku za znacznym wzrostem opłat za przesyłki pocztowe, mając na uwadze dobro naszych czytelników sugerujemy zamawianie numerów zaległych naszego czasopisma na nieco innych zasadach.

Koszty przesyłki za tzw. zaliczeniu pocztowym przedstawia tabela (rubryki 2, 3, i 4). Wynika z niej, że koszty przesłania dwóch pierwszych numerów „64 plus 4" są większe niż ich wartość! Zamawiając numery wartości 6.000 zł zmuszeni jesteście dopłacić poczcie jeszcze 8.000 zł!

Zupełnie inaczej przedstawia się sytuacja w tabelach 5, 6 i 7. Wpłata kwoty z rubryki nr 7 na nasze konto (wraz z czytelną adnotacją, których numerów dotyczy, umieszczoną na wszystkich odcinkach blankietu) powoduje, że otrzymujecie przesyłkę bez kosztów pobrania!

Proponujemy abyście zaległe numery naszego pisma zamawiali w sposób następujący: wykorzystując tabelę - rubryki 1, 2, 5 i 7 - wyliczyli kwotę wpłaty, a następnie przesłali ją na nasze konto. Po otrzymaniu wpłaty natychmiast realizujemy przesyłkę!

Uwaga: dla dokonania wpłaty prosimy wykorzystać blankiet zamieszczony na następnej stronie. Wszystkie dane prosimy pisać czytelnie i - zgodnie z rubrykami na blankiecie - **dziękujemy!**

Numery	Za pobraniem			Wpłata na konto		
	Wart. pisma	Koszt wysyłki	Razem	Wart. pisma	Porto	Razem
1	2	3	4	5	6	7
XI,XII	6.000,-	8.000,-	14.000,-	6.000,-	1.000,-	7.000,-
XI,XII,I	11.000,-	8.500,-	19.500,-	11.000,-	1.500,-	12.500,-
XI,XII,I,II	16.000,-	8.500,-	24.500,-	16.000,-	1.500,-	17.500,-
XI,XII,I,II,III	21.000,-	8.500,-	29.500,-	21.000,-	1.500,-	22.500,-
XI,XII,I-IV	26.000,-	10.000,-	36.000,-	26.000,-	2.000,-	28.000,-
XI,XII,I-V	31.000,-	10.000,-	41.000,-	31.000,-	2.000,-	33.000,-
XI,XII,I-VI	36.000,-	10.000,-	46.000,-	36.000,-	2.000,-	36.000,-
XI,XII +I-VIII	46.000,-	14.000,-	60.000,-	46.000,-	3.000,-	49.000,-

WSZYSTKICH ZAINTERESOWANYCH NABYCIEM ZALEGŁYCH NUMERÓW

„64 plus 4 & AMIGA"

INFORMUJEMY, ŻE POSIADAMY JESZCZE OGRANICZONĄ ILOŚĆ NUMERÓW OD LISTOPADA 1990R. DO CZERWCA 1991R.

ZAMÓWIENIA PROSIMY KIEROWAĆ NA ADRES :
Przedsiębiorstwo ABUK sp. z o.o.,
87-200 Wąbrzeźno,
ul. 1 Maja 33.

(Pod tym adresem mieści się dział kolportażu - tam też prosimy przesyłać wszelką korespondencję dotycząca kolportażu czasopisma, dyskietek, taśm itd. Adres redakcji się nie zmienił - patrz stopka.)

ZAMÓWIONE NUMERY PRZEŚLEMY ZA ZALICZENIEM POCZTOWYM.

W związku z pojawiającymi się kłopotami w dystrybucji oferowanych przez nas dyskietek i taśm (wynikającymi z nieczytelnego bądź niekompletnego wypełnienia blankietów wpłat) przedstawiamy obok specjalny druk. Blankiet ten może służyć jako zamówienie i dowód wpłaty dla wszystkich oferowanych przez nas usług: sprzdaż dyskietek i taśm PDP, Voicetracker'a, zamówienie ogłoszeń itd.

REDAKCJA

UWAGA!

W związku ze wzrostem opłat pocztowych (nasze pismo jest już grubsze i cięższe, a więc musimy za jego wysyłkę płacić więcej)

z przykrością informujemy, że od 1 sierpnia cena 1 egzemplarza „64 plus 4" w prenumeracie wynosi 5000zł.

Przypominamy, że prenumeratę można zawrzeć na dowolny okres (nie krótszy niż dwa miesiące) do końca bieżącego roku. Na **CZYTELNIE** wypełnionych blankietach wpłat prosimy dopisać **„PRENUMERATA" oraz okres jej trwania.** Wpłaty prosimy przesyłać wykorzystując zamieszczony obok blankiet. Przepraszamy.

Odcinek dla wpłacającego

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla Banku

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla posiadacza rachunku

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla Poczty

Zł ....
słownie ....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-100 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

TREŚĆ ZAMÓWIENIA:

....

TREŚĆ ZAMÓWIENIA:

....

Prosimy ■ CZYTELNE wypełnienie

TREŚĆ ZAMÓWIENIA:

....

Prosimy ■ CZYTELNE wypełnienie.

PAMIĘTAJCIE !

Czytelne i dokładne wypełnienie wszystkich rubryk gwarantuje szybką i poprawną realizację Waszego zamówienia!

DZIĘKUJEMY!

STATEX
PRACOWNIA KOMPUTEROWA

01-911 Warszawa, ul. Andersena 2

oferuje

PEŁNY SERWIS SPRZĘTU COMMODORE 64 / AMIGA, PC - XT/AT,
stacje dysków, drukarki, cartridge.

Firma

KOMPART

BYDGOSZCZ
ul. Poznańska 19

oferuje:

- naklejki na papierze i foliach samoprzylepnych (również na dyskietki);
- litery z folii samoprzylepnej wycinane ploterem (48 krojów pisma);
- materiały poligraficzne, folie i papiery samoprzylepne;
- literaturę i czsopisma komputerowe (wszystkie numery 64 plus 4);
- programy na Amigę, C-64 (również dyski PDP i VOICETRACKER V4.0), Atari XE/XL;
- naprawy sprzętu komputerowego.

Z A P R A S Z A M Y !

COMMODORE 64
AMICOS COMPUTER
42-200 CZĘSTOCHOWA
TEL 22-22-38
ATRAKCYJNE PROGRAMY NA KASETACH
CIEKAWE PROGRAMY NA DYSKACH
KATALOGI GRATIS (koperta+znaczek)
WYSYŁKA POCZTĄ EKSPRESOWA



MIKRO SERWIS
80-288 GDAŃSK MORENA
ul.Marusarzówny 6
tel 48 50-63 900-1700

Oferujemy do komputera AMIGA 500
ROZSZERZENIE RAM!

do 1 MB
do 2.3 MB
do 2.5 MB



Wszystkie rozszerzenia mogą być wyposażone w zegar z podtrzymaniem akumulatorowym.
Prowadzimy też naprawy sprzętu komputerowego i peryferii.

SCULPT ANIMATE 4D cz. 4 (Menu Tools i Edit)

W dzisiejszym odcinku naszej opowieści o Sculpt'cie omówię kilka nowych rzeczy. Na pierwszy ogień pójdzie menu Tools, które składa się z kolejnych pozycji: Selector, Deselector, Magnet, Curve, Extrude, Edge maker, Grabber.

SELECTOR

Po wybraniu opcji Selector otrzymamy zamiast kursora mały prostokącik, którym możemy dokonywać wyboru większej ilości punktów. Wszystkie punkty znajdujące się w obrębie tego prostokącika w czasie, gdy klikamy myszą zostaną wybrane (zmieni się ich kolor na żółty).

DESELECTOR

Wybierając opcję Deselector otrzymujemy prostokącik koloru różowego, dzięki któremu możemy deaktywować wszystkie wcześniej uaktywnione (żółte) punkty. Punkt zdeaktywowany ma kolor różowy i nie można wykonywać na nim żadnych operacji tj. przenoszenia, usuwania czy tworzenia krawędzi (edge).

MAGNET

Opcja: Magnet

Do wyboru: Attract, Repel
Są to dwa zwykłe magnesiki: Attract (Przyciągający) i Repel (Odpychający), z których wybieramy jeden aktualnie nam potrzebny i po zmianie kursora na mały magnesik wciskamy lewy przycisk myszki. Trzymając go cały czas wciśnięty, przesuwamy magnesik w odpowiednie miejsce i wciskamy prawy przycisk myszy. Punkty aktywne będą się zbliżać, lub oddalać w zależności od użytego magnesu. Aby opuścić opcję Magnet należy zwolnić lewy przycisk myszy, a następnie wcisnąć prawy.

CURVE

Za pomocą opcji Curve możemy tworzyć figury o dowolnie wybranych kształtach. Po wybraniu tej opcji zmienia nam się wygląd kursora. Tworzenie figury odbywa się na zasadzie rysowania kolejnych punktów łączonych liniami. Ustawiamy kursor w miejscu, gdzie chcemy rozpocząć nasze dzieło i wciskamy lewy przycisk myszy. Następnie trzymając wciśnięty przycisk, przesuwamy kursor w miejsce gdzie chcemy mieć drugi punkt tej figury i przyciskamy prawy przycisk myszki. Zostanie narysowana linia łącząca dwa powstałe punkty. Teraz możemy powtarzać tę operację aż do znudzenia. Gdy już figura jest gotowa należy opuścić opcję Curve, a można dokonać tego w ten sam sposób, w jaki opuszcza się większość opcji Sculpta, to jest poprzez wciśnięcie prawego przyciku naszej myszki.

EXTRUDE

Extrude jest bardzo pomocną opcją zwłaszcza przy tworzeniu obiektów, np. napisów, które należy pogrubić względem jednej z osi. Uaktywniamy odpowiednie punkty figury i wybieramy pozycję Extrude. Chwila oczekiwania i oto mamy... Pozornie nic się nie zmieniło, ale to tylko złudzenie. Nasza figura została pogrubiona, ale jest ściśnięta i teraz należy przesunąć jej tylną ściankę w odpowiednie miejsce, aby nadać naszej figurze właściwą grubość. Przesunięcia dokonujemy za pomocą opcji Grabber, do której wchodzimy automatycznie po opuszczeniu opcji Etrude, ale o tym za chwilę.

EDGE MAKER

Tworzy krawędź pomiędzy dwoma punktami. Po wybraniu tej opcji wciskamy i przytrzymujemy lewy przycisk myszy, i przesuwamy kursor na odpowiedni punkt, a następnie wciskamy prawy przycisk. W tym momencie, od tego punktu będzie się ciągnąć kreska (nasza krawędź), którą prowadzimy do następnego punktu i tam wciskamy prawy przycisk myszy.
Wyjście z tej opcji następuje po wciśnięciu tylko prawego przycisku myszy.

GRABBER

Grabber jest urządzeniem do przenoszenia aktywnych punktów. Najpierw ustawiamy kursor w odpowiednim miejscu, a następnie wybieramy opcję Grabber (można ją wybrać także poprzez kliknięcie na gadżecie w lewym dolnym rogu okienka edycji). Teraz wciskamy i przytrzymujemy lewy przycisk myszki, i przesuwamy kursor, wraz z nim są przesuwane aktywne punkty. Po przesunięciu do właściwej pozycji, wciskamy prawy przycisk aby opuścić tę opcję.

Kolejnym menu, które omówię jest menu Edit. Składa się ono z następujących pozycji: Select, Deselect, Erase, Modify, Do, Add, Snap, Name, Grid, Coordinates.

SELECT

Opcja: Select

Do wyboru: All onected, Indicated vertex, Swap, Named vertices, Indicated edge, Indicated spline, Indicated path.

Opcja Select służy do uaktywnienia poszczególnych punktów naszej sceny. Wybierając poszczególne pozycje z menu możemy dokonywać aktywacji różnych elementów:

ALL - wszytkie punkty ze sceny zostaną uaktywnione.

CONNECTED - wszystkie punkty połączone z punktem znajdującym się bezpośrednio pod kursorem zostaną uaktywnione.

INDICATED VERTEX - uaktywniony zostanie punkt (wierzchołek) znajdujący się pod kursorem.

SWAP - wszystkie aktywne (żółte) wierzchołki zostaną zdeaktywowane (różowe), a wszystkie nieaktywne zostaną uaktywnione.

NAMED VERTICES - aktywacja wszystkich nazwanych (będziemy zapytani o nazwę) wierzchołków.

INDICATED EDGE/SPLINE/PATH - wszystkie wierzchołki (punkty), krawędzie, ścieżki, itp. znajdujące się w obrębie kursora.

DESELECT

Opcja: Deselect

Do wyboru: All, Conected, Indicated vertex, Swap, Named vertices, Indicated edge.

Opcja Deselect działa odwrotnie do opcji select co oznacza, że deaktywuje ona wierzchołki (punkty).

ERASE

Opcja: Erase

Do wyboru: Selected vertices, Selected Edges, Indicated vertex , Indicated edge, Indicated lamp, All lamps, Named vertices, Indicated spline, Indicated path, All.

Opcja Erase służy do usuwania zbędnych elementów: punktów, krawędzi, lamp, itp.

ALL - wszysko zostanie usunięte (lampy, wierzchołki, itd.); wybieramy przy tworzeniu nowej sceny.

SELECTED VERTICES - wszystkie aktywne (żółte) wierzchołki zostaną usunięte.

NAMED VERTICES - zostaną usunięte wierzchołki o podanej nazwie.

INDICATED VERTEX - usunięty zostanie punkt znajdujący się pod kursorem.

SELECTED EDGES - wszystkie aktywne krawędzie zostaną usunięte.

INDICATED EDGE - zostanie usunięta krawędź znajdująca się pod kursorem.

ALL LAMPS - wszystkie lampy (źródła światła) zostaną usunięte.

INDICATED LAMP - zostanie usunięta lampa znajdująca się bezpośrednio pod kursorem.

MODIFY

Opcja: Modify

Do wyboru: Faces, Lamps, Wire-frame colors Local origin, Indicated knot, Indicated tumble Take.

Służy do zmiany parametrów dla poszczególnych elementów.

FACES - służy do zmiany parametrów ścian (metal, szkło, lustro, itp.) oraz ich kolorów.

LAMPS - służy do modyfikacji źródeł światła, ich natężenia, oraz koloru.

WIRE FRAME COLORS - kolory przy kreśleniu w trybie wire-frame.

INDICATED KNOT/THUMBLE - służy do modyfikacji parametrów animacji.

DO

Opcja: Do

Do wyboru: Expand, Subdivide, Spin, Fill Unslice, Reflect, Make helix, Make spline, Make path, Show path position, Hide selected verices, Reveal hidden vertices, Make Tri-View small, Make Tri-View big.

Opcja Do jest pomocniczą opcją i składa się na nią wiele różnych elementów.

EXPAND - wszystkie aktywne wierzchołki będą wykorzystane w tej opcji, a służy ona do powiększania, lub pomniejszania figur. Operacja odbywa się względem kursora.

SUBDIVIDE - dzieli wszystkie krawędzie pomiędzy aktywnymi wierzchołkami na dwie części poprzez wprowadzenie dodatkowych wierzchołków w połowie długości tych krawędzi.

SPIN - tworzy figurę obrotową z aktywnych wierzchołków.

FILL - służy do utworzenia ściany z aktywnych wierzchołków. Płaszczyzna wyznaczona przez nie będzie podzielona na trójkąty.

REFLECT - wykona lustrzane odbicie wszystkich aktywnych wierzchołków.

MAKE HELIX/PATH/SPLINE - służą do tworzenia animacji.

HIDE SELECTED VERTICES - schowa aktywne wierzchołki.

REVEAL HIDDEN VERTICES - ukaże wszystkie ukryte, poprzednią opcją, wierzchołki.

MAKE TRI-VIEW SMALL/BIG - ustawienie wielkości okien na których pracujemy.

ADD

Opcja: Add

Do wyboru: Duplicate, Sphere, Hemisphere, Cube, Prism, Disk, Circle, Cylinder, Tube, Cone, Lamp, Vertex, Edges.

Opcja Add pozwala na wprowadzenie już gotowych figur do naszej sceny.

DUPLICATE - tworzy figurę bliźniaczo podobną do figury utworzonej przez aktywne wierzchołki.

SPHERE - tworzy kulę (możemy wybrać ilość podziałów, czyli dokładność naszej kulki).

HEMISPHERE - tworzy pókulę (coś na kształt obserwatorium - wygląda bardzo ładnie!).

CUBE - zwykły (o ile ktoś zna niezwykłe!) sześcian.

PRISM - ostrosłup czworokątny.

DISK - taki talerzyk - podajemy liczbę wierzchołków (bo nie jest to koło, ale wielokąt foremny).

CIRCLE - tworzy okrąg.

CYLINDER - walec.

TUBE - rura.

CONE - stołek.

LAMP - i mamy kolejne źródło światła na pozycji kursora.

VERTEX - tam gdzie znajduje się kursor pojawi się wierzchołek.

EDGE - dwa aktywne wierzchołki zostaną połączone krawędzią.

SNAP

Opcja: Snap

Do wyboru: Cursor to vertex, Cursor to center, Cursor to centroid, Cursor to grid, Selected vertices to grid, Connected vertices to sphere, Selected vertices to plane.

Opcja Snap służy do ustawiania wybranych elementów na odpowiednich pozycjach.

CURSOR TO VERTEX - przesunie kursor do najbliższego wierzchołka.

CURSOR TO CENTER - przesunie kursor do środka okienek.

CURSOR TO CENTROID - kursor zostanie przesunięty do punktu, który jest średnią arytmetyczną pomiędzy wszystkimi wierzchołkami.

CURSOR TO GRID - przesunie kursor do najbliższego oczka siatki.

SELECTED VERTICES TO GRID - aktywne wierzchołki zostaną przesunięte do najbliższych im oczek siatki.

CONNECTED VERTICES TO SPHERE - połączone wierzchołki zostaną ułożone na kuli.

SELECTED VERTICES TO PLANE - aktywne wierzchołki zostaną sprowadzone do wspólnej płaszczyzny.

NAME

Opcja: Name

Do wyboru: Selected vertices, Connected vertices, Indicated path, Indicated lamp.

Name służy do nazywania elementów sceny, lub zmiany istniejących już nazw.

SELECTED VERTICES - umożliwia nazwanie wybranych (aktywnych) wierzchołków.

CONNECTED VERTICES - dzięki tej opcji możemy nazwać kilka połączonych ze sobą wierzchołków. Na przykład definiujemy kulę i ustawiamy kursor ponad jednym z jej wierzchołków. Po wybraniu tej opcji możemy nazwać wszystkie połączone wierzchołki tej kuli. Możemy im nadać np. nazwę KULKA i gdy chcemy się odwołać do wszystkich wierzchołków kuli (np. wymazać) to nie robimy żadnych operacji w stylu aktywne wierzchołki i tym podobne, ale po prostu usuwamy wierzchołki o nazwie KULKA.

INDICATED PATH/LAMP - umożliwia usunięcie lampy/ścieżki nad którą znajduje się kursor.

GRID

Opcja: Grid

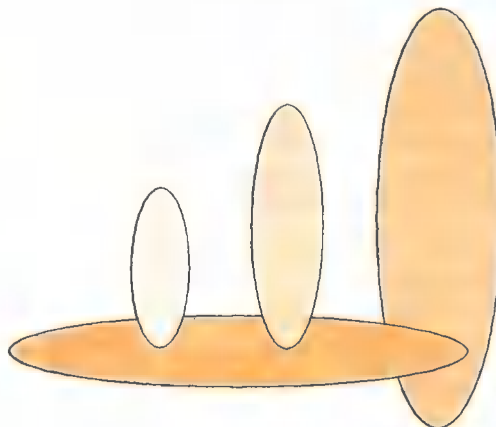
Do wyboru: On, Off

Opcja Grid służy do włączenia (On), lub wyłączenia (Off) siatki. Siatka jest to coś w rodzaju papieru milimetrowego co ułatwia dokładne ustawianie wierzchołków, oraz ułatwia orientację.

COORDINATES

Włączając opcję Coordinates, Sculpt 4D pokaże małe okienko, w którym będą wyświetlane pozycje kursora w definiowanej scenie.

Marcin „Duddle" Dudar



JAK ZROBIĆ WŁASNE DEMO (3)
Sprites

W dzisiejszym odcinku zapoznamy się z najprostszą techniką animacji obrazu: z wykorzystaniem SPRITE'ów. SPRITE są to obiekty o ograniczonej liczbie kolorów (do czterech uwzględniając kolor tła), szerokości 16 punktów i dowolnej wysokości. Dzięki temu, iż posiadają odrębne kanały DMA mogą być poruszane niezależnie od zawartości ekranów graficznych, co czyni je bardzo wygodnymi w obsłudze i sprawia, że znajdują szerokie zastosowanie.

Aby umieścić naszego SPRITE'a na ekranie musimy sporządzić specjalną strukturę, która poinformuje komputer o kształcie naszego obiektu i jego pozycji na ekranie. Struktura ta spełni podobne zadanie, jak Copper List dla procesora obrazu. Jednak zanim nauczymy się ją tworzyć, przedstawię w jaki sposób komputer interpretuje liczby będące wskaźnikami pozycji oraz dane dla grafiki, czyli kształtu SPRITE'a.

Pozycję SPRITE'a na ekranie charakteryzują dwie liczby - jego współrzędne: pozioma X, oraz pionowa Y. Są one względnym przesunięciem naszego obiektu od lewego, górnego rogu ekranu.

* Pozycja pozioma - X.

Pozycja pozioma SPRITE'a może być liczbą mieszczącą się w zakresie od 0 do 447. Jednakże, aby obiekt był widoczny na ekranie przy standardowo ustawionych borderach, jego współrzędna X nie powinna być mniejsza niż 64 i większa od 368. Ostatecznie mamy więc 368-64=304 dozwolonych pozycji poziomych. Uwzględniając, iż SPRITE może mieć maksymalnie 16 punktów szerokości, otrzymujemy 304+16=320, czyli tyle punktów iloma dysponujemy w trybie „Lo-Res".

* Pozycja pionowa - Y.

Pozycja pionowa może przyjmować wartości z zakresu 0 do 262. Zazwyczaj jednak używa się wartości z zakresu 44 do 243, co uwzględniając linię 0 daje nam 200 pozycji w pionie. Należy tu zaznaczyć, iż podany wyżej zakres dotyczy SPRITE'a o wysokości jednej linii,
a takich nikt nie będzie raczej używać (może jedynie do robienia gwiazdek). W przypadku używania objektów większych rozmiarów dolna granica (tzn. 243) odpowiednio zmniejszy się o wartość wysokości SPRITE'a.

Teraz, gdy poznaliśmy ograniczenia i możliwości umieszczania SPRITE'ów w różnych miejscach ekranu, zajmijmy się ich kształtem. Pierwszą rzeczą którą musimy zrobić jest wykonanie projektu SPRITE'a, najlepiej na papierze. Musimy przy tym pamiętać, że może mieć on jedynie 16 punktów szerokości i być w 3 kolorach + border. Załóżmy, że interesuje nas następujący kształt:

```
1111111111111111
1122222222222211
1122333333332211
1122330000332211
1122330000332211
1122333333332211
1122222222222211
1111111111111111
```

gdzie cyfry oznaczają:
1- kolor czerwony
2 - kolor pomarańczowy
3 - żółty
0 - kolor tła

Gdy kształt jest gotowy, rozkładamy nasz rysunek na dwa Bit-Plane'y, pamiętając o tym, że: 0=%00, 1=%01, 2=%10, 3=%11. W rezultacie otrzymujemy:

Bit-Plane 0	Bit-Plane 1
1111111111111111	0000000000000000
1100000000000011	0011111111111100
1100111111110011	0011111111111100
1100110000110011	0011110000111100
1100110000110011	0011110000111100
1100111111110011	0011111111111100
1100000000000011	0011111111111100
1111111111111111	0000000000000000

Przepisując teraz liniami poziomymi na przemian dane dla dwóch Bit-Plane'ów otrzymamy gotowe informacje dla kształtu SPRITE'a. W naszym przykładzie będą one przedstawiały się następująco:

```
dc.l %1111111111111111, %0000000000000000
dc.l %1100000000000011, %0011111111111100
dc.l %1100111111110011, %0011111111111100
dc.l %1100110000110011, %0011110000111100
dc.l %1100110000110011, %0011110000111100
dc.l %1100111111110011, %0011111111111100
dc.l %1100000000000011, %0011111111111100
dc.l %1111111111111111, %0000000000000000
```

Jeśli komuś wydaje się zbyt żmudne przepisywanie liczb binarnych, może zamienić je hexadecymalnymi. Dane przygotowane w ten sposób mogą być bezpośrednio wykorzystane w strukturze podawanej komputerowi.

Krzysztof "K.K." Kobus

Kącik początkującego kodera (cz. 6)

Dzisiaj, na początek nowa garść rozkazów, a potem krótki program.

Tym razem będzie to procedura służąca do sporządzania tablic adresów font NOP "No Operation" - nie rób nic.

Po napotkaniu tej instrukcji mikroprocesor nie wykonuje żadnego działania. Laikom mogłoby się wydawać, że instrukcja ta jest zupełnie niepotrzebna -przecież ona nic nie robi, a jednak jest przydatna i może być używana na wiele różnych sposobów.

Hackerzy zastępują rozkazy zmniejszania zawartości komórki zawierającej ilość „żyć" w grach instrukcjami NOP, tym samym uzyskując nieśmiertelność. Przy uruchamianiu programów można tą instrukcją zastąpić niektóre rozkazy, (np. skok do podprogramu), aby zobaczyć jak w takiej sytuacji zachowa się nasz program. Wreszcie może być wprowadzona do programu w celu uzyskania niewielkich opóźnień.

Znaczniki: nie są modyfikowane.

Przykład: NOP

DBxx Dx,ykieta "Test Condition, Decrement And Branch" - testuj warunek zmniejsz i skocz.

Za pomocą instrukcji DBxx możemy w łatwy sposób organizować pętle. W tym celu należy wybrać jeden z re jestrów danych, który spełni zadanie licznika, po czym zainicjować go odpowiednią wartością. Następnie początek pętli oznaczymy dowolną etykietą (np. „Loop"), a na jej końcu umieścimy jedną z instrukcji DBxx. Mikroprocesor po napotkaniu tejże komendy sprawdzi określone warunki (zależnie od „xx") i jeżeli będą one spełnione, zakończy wykonywanie pętli, przechodząc do następnego rozkazu. Jeżeli natomiast dany warunek nie będzie spełniony, mikroprocesor zmniejszy o 1 zawartość wybranego rejestru danych, czyli naszego licznika i przejdzie do etykiety, o ile licznik nie osiągnie wartości -1. W tym ostatnim przypadku pętla zostanie przerwana. Należy pamiętać, że rozkazy DBxx operują jedynie na młodszych 16-bitach rejestrów, czyli na młodszym słowie. Przy organizowaniu pętli proponuję korzystać z wyższych rejestrów danych tzn. d6 lub d7, ponieważ są one rzadziej używane, a to pozwoli uniknąć błędów. Przedstawię teraz pokrótce wszystkie instrukcje DBxx.

```
DBCS :pętla    zostanie    zakończona    jeśli C=1
DBCC :pętla    zostanie    zakończona    jeśli C=0
DBEQ :pętla    zostanie    zakończona    jeśli Z=1
DBNE :pętla    zostanie    zakończona    jeśli Z=0
DBMI :pętla    zostanie    zakończona    jeśli N=1
DBPL :pętla    zostanie    zakończona    jeśli N=0
DBVS :pętla    zostanie    zakończona    jeśli V=1
DBVC :pętla    zostanie    zakończona    jeśli V=0
DBGE :zakończenie, gdy [N=0 i V=0], lub [N=1 i V=1]
DBGT :zakończenie, gdy [N=1 i V=1 i Z=0], lub [N=0 i
     ;V=0 i Z=0]
DBHI :zakończenie, gdy C=0 i Z=0
DBLE :zakończenie, gdy [Z=1] lub [N=1 i V=0] lub
     ;[N=0 i V=1]
DBLS :zakończenie, gdy C=1 albo Z=1
DBLT :zakończenie, gdy [N=1 i V=0] lub [N=0 i V=1]
DBRA :nie testuje żadnych znaczników; pętla może być
     ;zakończona jedynie przez licznik. Mnemonik DBRA
     ;używany jest zamiennie z DBF.
DBT  :pętla nie zostanie powtórzona ani razu.
```

Nie zapominajmy, że w każdym z wyżej wymienionych przypadków pętla może być również zakończona, gdy licznik osiągnie wartość -1.

Znaczniki: nie są modyfikowane.

Przykład:

```
       move.w   #$13,d7
loop:  move.b   (a0)+,(a1)+
       dbeq     d7,loop
```

Wykonanie tych instrukcji da nam następujący rezultat:

bajty będą przepisywane spod adresu wskazywanego przez a0, pod adres wskazywany przez a1.
Jeżeli jeden z nich będzie równy 0, pętla zostanie przerwana.

W przypadku, gdy wszystkie będą różne od 0, pętla zostanie powtórzona $14 razy, gdy $13-$14=-1.

OR adres efektywny,Dx „Inclusive OR Logical" - Logiczna suma Dx,adres efektywny.

Instrukcja OR wykonuje logiczną sumę dwóch operandów zostawiając wynik w operandzie przeznaczenia. Sumą logiczną nazywamy funkcję, która przyjmuje wartość 1, jeżeli przynajmniej jeden z argumentów jest równy 1. Możliwe przypadki ilustruje tabelka:

0 lub 0 = 0
0 lub 1 = 1
1 lub 0 = 1
1 lub 1 = 1

Instrukcji tej używamy, gdy chcemy ustawić w operandzie określone bity. Operować możemy na bajtach, słowach i długich słowach.

Znaczniki: X - bez zmian
N - ustawiany zgodnie z najstarszym bitem ;wyniku
Z - ustawiany, gdy wynik jest równy zero
V - zawsze zerowany
C - zawsze zerowany

Przykład:

```
       move.b   #%01110100,d0
       move.b   #%11000010,d1
       or.b     d0,d1
```

W wyniku otrzymamy wartość %11110110 umieszczoną w rejestrze d1.

ORI #liczba,adres efektywny „Inclusive OR Immediate" - natychmiastowe OR.

Instrukcja ta wykonuje logiczną sumę natychmiastowego operandu źródłowego z operandem przezna-

czenia określonym adresem efektywnym. Możliwe jest wykonanie ORI z rejestrem statusu mikroprocesora, co pozwola programiście na dowolne ustawienie znaczników. Rozmiarem tej operacji może być bajt, słowo lub długie słowo.

Znaczniki: analogicznie jak OR.

W przypadku wykonania ORI z rejestrem stanu znaczniki zostaną ustawione zgodnie z operandem źródłowym.

Przykład:

```
ori.b    #$23,d0 - wykona logiczne or z bajtem
         ;rejestru d0.
```

EXG Rx,Ry "Exchange Registers" - zamiana rejestrów.

Instrukcja EXG służy do zamiany dwóch rejestrów. Możliwe są trzy przypadki:

-zostaną zamienione zawartości rejestrów danych

-zostaną zamienione zawartości rejestrów adresowych

-zamiana rejestru danych z adresowym.

Zawsze zamieniane są wszystkie 32-bity - nie można zamienić np. tylko młodszych słów rejestrów.

Znaczniki: nie są modyfikowane.

Przykład:

```
exg a0,d5 - wymieni zawartość rejestrua0 z d5.
```

NOT adres efektywny „Logical Complement" - dopełnienie logiczne.

Instrukcja ta neguje operand określony adresem efektywnym. Stan wszystkich bitów zmieni się na przeciwny. Logiczne NOT można wykonywać na operandzie wielkości bajtu, słowa lub długiego słowa.

Znaczniki: X - nie zmieniany
N - ustawiany, gdy wynik jest ujemny
Z - ustawiany, gdy wynikiem jest 0
V - zawsze zerowany
C - zawsze zerowany.

Przykład:

```
move.b   #%11100011,d1
not.b    d1.
```

W wyniku działania tych rozkazów otrzymamy liczbę %00011100, umieszczoną w rejestrze d 1.

Jak już wspomniałem na wstępie, procedura, którą dzisiaj przedstawię służy do sporządzania tablic adresów fontów lub elementów grafiki. Jeżeli komuś wydaje się to niezrozumiałe, proszę o rozpatrzenie poniższego przykładu.

Załóżmy, że piszemy grę „komnatową" - musimy, więc na jednym dysku zmieścić ogromną ilość grafiki. Aby to umożliwić stosuje się następującą metodę: grafik nie rysuje całych komnat, tylko małe ich fragmenty (o stałych wymiarach np: 16x16). Następnie programista z tych małych klocków składa, na podstawie wcześniej ustalonych tablic, całe komnaty. Pozwala to na ogromne zaoszczędzenie pamięci, gdy jeden klocek może wielokrotnie występować w danej komnacie. Podobnie postępują koderzy. Aby zaoszczędzić czas, nie obliczają za każdym razem pozycji fonta w zestawie, lecz pobierają ich adresy z tablicy sporządzonej przy pomocy procedur podobnych do powyższej.

Myślę, że znajdzie ona szerokie zastosowanie, a na pewno warto ją przeanalizować ze względów treningowych. Dodam jeszcze, iż była to pierwsza procedura, jaką napisałem w asemblerze 68000.

Zapoznajcie się z jej działaniem i spróbujcie ją napisać od nowa lub przynajmniej zoptymalizować.

```
fonts      =$6000   ;adres pierwszego fonta w zestawie
lx:        =10      ;ilość fontów w linii
vx:        =4       ;szerokość fonta w bajtach
vy:        =32      ;wysokość fonta
ll:        =5       ;liczba fontów w zestawie.
Start:
       clr.l    d2
       lea      Tab(pc),a5
       move.l   #Fonts,d3
       move.b   #ll,d7
       bsr      Wstaw
       move.b   #1,d6
Loop:  addi.l   #vx,d3
       subq.b   #1,d7
       beq      End
       addq.b   #1,d6
       bsr      Wstaw
       cmp.b    #lx,d6
       bne      Loop
       clr.l    d0
       clr.b    d6
       move.b   #[vy-1],d0
       mulu     #lx*vx,d0
       add.l    d0,d3
       bra      Loop
       End:     rts
Wstaw:move.l    d3,(a5)+
       rts
Tab:   blk.l    51,0          ;bufor na adresy fontów.
```

Krzysztof "K.K." Kobus

FILE REQUESTE
czyli o wybieraczkach słów kilka

Często pisząc jakiś program, który będzie się komunikował z dyskiem, napotykamy na problem eleganckiego, przyjemnego dla użytkownika, a jednocześnie funkcjonalnego wybierania zbioru z katalogu dysku. Wiele programów (np. Cygnus Edytor v1.0 czy Power Packer) ma swoje własne programy obsługi katalogu dysku, jednak są to dosyć długie i raczej skomplikowane procedury. Cóż więc należy uczynić, aby nasz (najczęściej krótki) program nie wydłużył się na „pół" pamięci, oraz żeby zaoszczędzić sobie pracy? Myślało już o tym wielu programistów i... stworzyli bardzo dobre „wybieraczki" (file requester'y), które są umieszczone w popularnych bibliotekach - ARP i REQ. Biblioteki arp.library (Amiga Replacement Project) i req.library (Requesters) są dostępne na wielu dyskach (np. „64+4 Public Domain").

Requestery te były używane w bardzo wielu programach, a oto niektóre przykłady:

Requester Arp:
- Resource
- ShowPic
- MasterSeka v1.80 - opcjonalnie
- PPShow 1.1
- PPMore 1.7

Requester Req:
- CED V2
- Noise Player v3.0
- MasterSeka v1.80 - opcjonalnie

Co należy uczynić, aby wykorzystać requester? Po pierwsze musimy posiadać daną bibliotekę na dysku, na którym pracujemy. Następnie musimy tą bibliotekę otworzyć i obsłużyć requester, a potem pobrać wybraną nazwę i robić z nią co tylko dusza zapragnie.

Zacznijmy więc od requestera z biblioteki arp.

Najpierw musimy się upewnić, że posiadamy bibliotekę arp.library na dysku, otwieramy ją i gdy chcemy wybrać jakiś zbiór, wykorzystujemy procedurę: FileRequest, która pobierze, i jako wynik da nam nazwę zbioru wybranego przez użytkownika.

Procedura:	FileRequest (arp.library)
Skok:	-294
Wejście:	A0 - adres struktury FileRequester
Wyjście:	D0 - wskaźnik nazwy zbioru w strukturze FileRequester, lub jeśli D0 = 0 to wybrano opcję „Cancel"

Struktura FileRequester

CPTR	(dc.l)	FR_Hail	adres nazwy okna
CPTR	(dc.l)	FR_File	adres obszaru przeznaczonego na nazwę zbioru
CPTR	(dc.l)	FR_Dir	adres obszaru przeznaczonego na nazwę katalogu
CPTR	(dc.l)	FR_Window	przyporządkowanie requestera dla danego okna lub 0 jeżeli dla Workbencha
UBYTE	(dc.b)	FR_FuncFlags	
UBYTE	(dc.b)	FR_reserved1	zarezerwowany (należy ustawić na 0)
APTR	(dc.l)	FR_Function	adres procedury wywoływanej dla WildCards'ów
LONG	(dc.l)	FR_reserved2	zarezerwowany (należy ustawić na 0).

Tak wygląda struktura FileRequester, a teraz po kolei o każdym z parametrów.

FR_Hail jest to adres nazwy okna, np.

Nazwa dc.b „Load File" 0; koniecznie musi kończyć się zerem.

FR_File - pod tym adresem będzie umieszczana nazwa wybranego zbioru przez użytkownika. Obszar ten powinien mieć długość 33 bajtów.

FR_Dir - pod tym adresem będzie umieszczana nazwa katalogu, z którego wybieramy dany zbiór. Obszar ten powinien mieć długość 34 bajtów.

FR_Function - adres procedury obsługującej np. ukrywanie niektórych zbiorów (np. .info). Gdy zostanie wykonany skok do tej procedury, to w D0 otrzymujemy maskę znacznika (FR_FuncFlags), dla którego została wywołana ta funkcja, a w A0 wskaźnik obiektu przyporządkowanego temu znacznikowi. Maska znacznika nie jest numerem bitu danego znacznika, ale wartością FR_FuncFlags dla tego znacznika. Rejestr adresowy A4 jest taki sam jak przed wejściem do procedury FileRequest, ale rejestr A6 nie musi wcale zawierać adresu biblioteki arp (tzw. ArpBase). Nasza procedura może dowolnie używać rejestrów A0, A1, D0, D1 oraz A4, natomiast pozostałe rejestry musimy przechować (łącznie z A6). Przy wyjściu dla niektórych znaczników wymagana jest maska w D0.

FR_FuncFlags - znaczniki sterujące wywołaniem procedury FR_Fuction. Jeżeli dany bit jest ustawiony to procedura zostanie wywołana z maską tego z danymi dotyczącymi danego bitu.

A teraz po kolei:

FRB_DoWildFunc bit 7.

W A0 otrzymujemy FileInfoBlock (patrz Examine - dos.library) i jeżeli chcemy, aby ta nazwa została dodana do listy nazw w oknie requestera, to w D0 zwracamy 0, jeżeli zwrócimy wartość różną od zera, to zbiór zostanie pominięty.

FRB_DoMsgFunc bit 6.

W A0 otrzymujemy wskaźnik IntuiMessage dla danego okna, z którego został wywołany requester. Możemy teraz wpływać na IntuiMessage, dotyczące okna i gadżetów mu przyporządkowanych. Wartość w D0 jest ignorowana.

FRB_DoColor bit 5.

Funkcja nie jest wywoływana, ale gdy ten bit jest ustawiony, kolor tła dla requestera jest inny dzięki czemu możemy mieć dwa różne requestery, np. jeden dla funkcji load, a drugi dla funkcji save.

FRB_NewIDCMP bit 4.

Używany tylko, gdy requester jest wywoływany z danego okna (FR_Window 0) wtedy requester pobierze znaczniki IDCMP z portu tego okna. Procedura nie jest wywoływana.

FRB_NewWindFunc bit 3.

W A0 otrzymujemy adres struktury NewWindow dla requestera, dzięki czemu możemy zmodyfikować wymiary oraz pozycję jego okna. Wartość w D0 jest ignorowana.

FRB_AddGadFunc bit 2.

W A0 otrzymujemy adres struktury Window dla requestera po tym, jak dodał wszystkie swoje gadżety, ale ich jeszcze nie wyrysował. Możemy w ten sposób dodać własne gadżety do okna requestera. Wartość w D0 jest ignorowana.

FRB_GEventFunc bit 1.

Procedura zostanie wywołana gdy, któryś z gadżetów użytkownika zostanie wybrany. W A0 otrzymujemy gadżet ID dla gadżetu wybranego przez użytkownika. Jeżeli w D0 zwrócimy 0 FileRequest nie zareaguje i będzie kontynuował, w przeciwnym wypadku zakończy działanie i wróci do głównego programu.

FRB_ListFunc bit 0.

Jeszcze nie został wykorzystany przez autorów arp.library.

A teraz krótki przykład wykorzystania requestera Arp.

```
Exec:               =     4
OldOpenLibrary:     =   - 408
FileRequest:        =   - 294
            move.l      Exec,a6
            lea         ArpName,a1
            jsr         OldOpenLibrary(a6)
            move.l      D0,ArpBase
            move.l      ArpBase,a6
            lea         FileRequester,a0
            jsr         FileRequest(a6)
            tst.l       d0
            beq.s       Error
            ;tu procedura wykorzystania nazwy
            rts
Error:      ;procedura obsługi błędu (użytkownik
            ;wybrał opcję CANCEL w requesterze)
            rts
ArpBase:
            dc.l        0
FileRequester:
            dc.l        Name
            dc.l        FileMem
            dc.l        DirMem
            dc.l        0       ;nie dołączamy do
                                ;żadnego okna
            dc.b        0
            dc.b        0
            dc.l        0       ;nie korzystamy
                                ;z żadnej procedury
            dc.l        0
ArpName:
            dc.b        "arp.library",0
Name:       dc.b        "Load File",0
FileMem:
            blk.b       33,0
DirMem:
            blk.b       34,0
```

Tak wygląda obsługa File Requestera z biblioteki arp, a teraz przejdźmy do „wybieraczki" z biblioteki req.

File Requester umieszczony w tej bibliotece jest wygodniejszy i posiada o wiele więcej funkcji niż ten z arp.

```
Procedura:  FileRequester (req.library) Skok: - 84
Wejście:    A0 - adres struktury FileRequester
Wyjście:    D0 - jeżeli D0 = 0 wybrano opcję Cancel
            ;w przeciwnym wypadku wybrano zbiór
```

Struktura FileRequester

UWORD(dc.w)VersionNumber ustawiamy na 0.

CPTR(dc.l)Title wskaźnik nazwy okna.

CPTR (dc.l) Dir adres obszaru przeznaczonego na nazwę katalogu obszar ten musi mieć długość 131 bajtów.

CPTR (dc.l) File adres obszaru przeznaczonego na nazwę zbioru, obszar ten musi mieć długość 31 bajtów.

CPTR (dc.l) PathName adres obszaru przeznaczonego na kompletną nazwę wybranego zbioru tj. katalog+zbiór, a jego długość musi wynosić 162 bajty.

CPTR (dc.l) Window adres struktury Window - podporządkowanie requestera do danego okna (jeżeli 0, to okno na Screen'ie Workbencha).

UWORD (dc.w) MaxExtendSelect maksymalna liczba zbiorów, które możemy wybrać jeżeli jesteśmy w trybie wybierania kilku zbiorów na raz (ustawiony znacznik FRQEXTSELECT) - 0 oznacza maksymalną liczbę zbiorów (65535).

UWORD (dc.w) NumLines ilość linii (zbiorów) wyświetlanych w oknie Requestera.

UWORD (dc.w) NumColumns ilość kolumn (szerokość) w oknie wybierania zbiorów

UWORD (dc.w) DevColumns szerokość okna, w którym są wyświetlane dostępne katalogi.

ULONG (dc.l) Flags znaczniki dla FileRequestera.

UWORD (dc.w) DirNamesColor znaczniki kolorów (wartości 1,2,3,...) jeżeli chcemy kolor 0, musimy wybrać wartość większą niż 32, 0 oznacza kolor narzucony przez req.

UWORD (dc.w) FileNamesColor.

UWORD (dc.w) DeviceNamesColor.

UWORD (dc.w) FontNamesColor.

UWORD (dc.w) FontSizesColor.

UWORD (dc.w) DetailColor jeżeli te dwa kolory są ustawione na 0 to

UWORD (dc.w) BlockColor będzie ustawiony kolor1

UWORD (dc.w) GadżetTextColor kolor pięciu gadżetów FileRequestera

UWORD (dc.w) TextMessageColor kolor komunikatu

UWORD (dc.w) StringNameColor kolor słów: Drawer,File,Hide,Show

UWORD (dc.w) StringGadgetColor Kolor ramek od gadżetów

UWORD (dc.w) BoxBorderColor kolor ramek wokół zbiorów

UWORD (dc.w) GadgetBoxColor kolor ramek wokół pięciu gadżetów

UWORD (blk.w) FRU_Stuff[18] 34 bajty zarezerwowane powinny być wyzerowane

STRCT (blk.b) DateStamp[14] kopia daty pobranej z katalogu (14 bajtów)

UWORD (dc.w) WindowLeftEdge

UWORD (dc.w) WindowTopEdge

te dwa pola wskazują położenie górnego lewego rogu okna requestera pod warunkiem, że jest ustawiony znacznik FRQABSOLUTEXY

UWORD (dc.w) FontYSize

UWORD (dc.w) FontStyle

w tych dwóch polach otrzymujemy wartości odnośnie fontów, jakie wybraliśmy pod warunkiem, że ustawiliśmy bit selekcji fontów FRQGETFONTS

APTR (dc.l) ESStructure jeżeli ustawiliśmy bit FRQEXTSELECT to w tym miejscu znajduje się wskaźnik do struktury ExtendedSelect (jeżeli użytkownik wybrał kilka zbiorów)

CHAR (blk.b) Hide[32] 32 bajty na maskę dla zbiorów, które będą ukrywane w katalogu

CHAR (blk.b) Show[32] 32 bajty na maskę dla zbiorów, które będą pokazywane w katalogu

WORD (dc.w) FileBufferPos

WORD (dc.w) FileDispPos

WORD (dc.w) DirBufferPoste

WORD (dc.w) DirDispPos FileRequestera

WORD (dc.w) HideBufferPos

WORD (dc.w) HideDispPos

WORD (dc.w) ShowBufferPos

WORD (dc.w) ShowDispPos

powyższe pola są pobrane z różnych gadżetów i wskazują pozycję kursora w tych gadżetach dzięki czemu przy kolejnych wywołaniach kursor będzie się znajdować w pozycji w jakiej go pozostawiliśmy ostatnio.

Kolejne pola są tylko do użytku wewnętrznego przez FileRequester!!! Nie należy ich modyfikować, gdy mogą z tego wyniknąć nieprzewidziane konsekwencje!!!

APTR (dc.l) Memory wskaźnik przestrzeni zaalokowanej na katalog.

APTR (dc.l) Memory2 wskaźnik pamięci użytej na zbiory ukrywane

APTR (dc.l) Lock

char (blk.b) PrivateDirBuffer[132] bufor zarezerwowany na przechowanie nazwy katalogu

APTR (dc.l) FileInfoBlock

WORD (dc.w) NumEntries

WORD (dc.w) NumHiddenEntries WORD (dc.w) FileStartNumber

WORD (dc.w) DeviceStartNumber

Tak wygląda kompletna struktura File Requester. Teraz przedstawię opis znaczników (Flags)

FRQSHOWINFO bit 0

Jeżeli ten znacznik jest ustawiony, to zbiory .info (Ikonki) będą pokazywane w katalogu. W przeciwnym przypadku zbiory te będą ukrywane.

FRQEXTSELECT bit 1

Jeżeli ten znacznik jest ustawiony, to użytkownik jest w trybie Extended, czyli może wybrać kilka zbiorów, a od naszego programu będzie zależało w jaki sposób zostanie to wykorzystane. W przeciwnym wypadku możemy wybrać tylko jeden zbiór.

FRQCACHING bit 2

W przypadku gdy ustawimy ten bit, FileRequester będzie rezerwował pamięć na katalog dysku i gdy uruchomimy go po raz kolejny, ten katalog będzie już wczytany i nie będzie musiał być za każdym razem wczytywany, gdy tylko wywołamy FileRequestera.

FRQGETFONTS bit 3

Jeżeli ustawimy ten bit, otrzymamy zupełnie inne okno FileRequestera służące do wybierania fontów (czcionek).

FRQINFOGADGET bit 4

Jeżeli ustawimy ten znacznik, to w prawym górnym rogu okna FileRequestera będzie umieszczony gadżet „Hide .info files" - ukrywanie zbiorów .info. Jeżeli przełączymy ten gadżet za pomocą myszy na „Show .info files" to zbiory .info pojawią się w katalogu.

FRQHIDEWILDS bit 5

Ustawiając ten znacznik pozbywamy się gadżetów „Hide" i „Show" służących do selekcji zbiorów posiadanych w katalogu. Jeżeli te gadżety są umieszczone w oknie FileRequestera to możemy aktywnie wpływać na filtrowanie plików, na przykład ustawiając „Show" na „*.s" będziemy otrzymywać w kata logu tylko te pliki, które posiadają dopełnienie „s".

FRQABSOLUTEXY bit 6

Jeżeli ustawimy ten bit, to okno FileRequestera będzie umieszczone na konkretnych pozycjach pobranych ze struktury FileRequester. W przeciwnym przypadku jest ono centrowane w stosunku do pointera myszy.

FRQCACHEPURGE bit 7

Jeżeli ten bit jest ustawiony to katalog będzie usunięty z pamięci, gdy tylko zmieni się data w głównym katalogu.

FRQNOHALFCACHE bit 8

Jeżeli ten bit jest ustawiony, to katalog zostanie zapamiętany tylko w przypadku, gdy został cały wczytany do pamięci. Jeżeli wczytywanie katalogu nie zostało ukończone to przy każdym wywołaniu FileRequestera będzie ten katalog wczytywany od początku.

FRQNOSORT bit 9

W przypadku ustawienia tego bitu katalog nie będzie sortowany przy wczytywaniu do pamięci.

FRQNODRAG bit 10

Jeżeli ustawimy ten bit, to okno FileRequestera nie będzie posiadało tzw. Drag Bar'a i gadżetów głębokości, czyli nie będziemy mogli wpływać na położenie tego okna względem innych okien, ani na jego położenie na ekranie.

FRQSAVING bit 11

Ustawiamy, gdy używamy FileRequestera do wybrania nazwy dla zbioru, który będziemy nagrywać na dysk.

FRQLOADING bit 12

Ustawiamy, gdy używamy FileRequestera do wybrania nazwy dla zbioru, który będziemy ładować do pamięci.

Bity FRQSAVING i FRQLOADING nie mają żadnego wpływu na pracę FileI equestera, i nie są do niczego wykorzystywane, powinny być ustawiane zgodnie z ich przeznaczeniem, gdyż może w przyszłych wersjach req.library znajdą zastosowanie.

I jeszcze tylko jedna struktura: ESStructure, która jest wykorzystywana gdy ustawimy bit FRQEXTSELECT.

Struktura ESStructure

APTR	(dc.l)	ESStructure	wskaźnik następnej struktury
WORD	(dc.w)	NameLength	długość nazwy zbioru
WORD	(dc.w)	Pad	tylko do wewnętrznego

```
                                  użycia
APTR    (dc.w)  Node              przez FileRequester
CHAR    (blk.b) TheFileName[1]    nazwa zbioru (zmienna).
```

Jeżeli chcemy wykorzystywać tryb Extended Select, to musimy mieć ustawiony także bit FRQCACHING, gdyż wtedy FileRequester tworzy i zapamiętuje struktury Extended Select.

I na koniec, przykład wykorzystania FileRequester'a z biblioteki req.library (musimy być w posiadaniu tej biblioteki i musi być ona umieszczona na dysku, na którym pracujemy).

```
Exec:                       =       4
OldOpenLibrary:             =       -408
FileRequester:              =       -84
            move.l          Exec,a6
            lea             ReqName,a1
            jsr             OldOpenLibrary(a6)
            move.l          D0,ReqBase
            move.l          ReqBase,a6
            lea             FileRequest,a0
            jsr             FileRequester(a6)
            tst.l           d0
            beq.s           Error
            ;tu procedura wykorzystania nazwy
            rts
Error:      ;procedura obsługi błędu (użytkownik
            ;wybrał opcję CANCEL w requesterze)
            rts
ReqBase:
            dc.l            0
FileRequest:
            dc.w            0
            dc.l            Name
            dc.l            DirMem
            dc.l            FileMem
            dc.l            PathMem     ;wskaźnik
                                        ;kompletnej nazwy
            dc.l            0
            dc.w            0
dc.w 0
            dc.w            0
            dc.w            0
            dc.l            $14         ;ustawione znaczniki
                                        ;FRQINFOGADGET
                                        ;i FRQCACHING
            blk.b           $164-[*-FileRequest],0
                                        ;struktura FileRequest
                                        ;ma długość $164
                                        ;bajtów i tyle pamięci
                                        ;musi być na nią
                                        ;zarezerwowane
ReqName:
            dc.b            "req.library",0
Name:       dc.b            "Load File",0
FileMem:
            blk.b           31,0
DirMem:
            blk.b           131,0
PathMem:
            blk.b           162,0
```

Na koniec, proponuję wczytać jakiś asembler i dokonać kilku eksperymentów - jest to najlepsza metoda na zrozumienie wszystkiego, co związane z programowaniem, gdyż nie można być programistą teoretykiem!

Marcin „Duddie" Dudar

BOOT X v3.50

Program BootX v3.50 nie należy może do nowości, ale gdy znalazłem go w swoim śmietniku programów (dyski na które nagrywam wszystko co się może kiedyś przydać) zdecydowałem, że powinien powędrować na pierwszy dysk z użytkami (zdarza się to bardzo rzadko ze względu na długa listę oczekujących i ze względu na to, że nieczęsto zdarza się, aby jakiś program opuścił Util Pack #1).

Autorem programu BootX v3.50 jest Peter Stuer (znany z wielu dobrych programów ShareWare). Program BootX został napisany całkowicie w assemblerze, co go zdecydowanie skraca i przyspiesza. Służy on do walki z wirusami i trzeba przyznać, że jest to narzędzie dobre, chociaż nie można powiedzieć, że jest doskonałe. Rozpoznaje on ponad 110 bootblock'ów (ponad 66 wirusów zpisujących się na bootblock'u), a także rozpoznaje i usuwa wirusy znajdujące się w zbiorach, lub linkujące się do innych programów.

Za pomcą BootX'a można prowadzić wojnę z takimi wirusami, znajdującymi się w zbiorach, jak: IRQ, BSG9, Lamer Exterminator, Xeno, Buttonic, CCCP, Travelling Jack oraz Revenge. Oprócz tych ponad 110 bootblockw, które BootX ma w sobie, możemy także tworzyć dla niego bibliotekę BootBlock'w, z której to będą pobierane wszystkie BootBlock'i, których nie ma on zdefiniowanych w sobie. Za pomocą BootX'a możemy nagrywać bootblock'i jako zbiory i odwrotnie: zbiory, jako bootblock'i. Program BootX pozwala na zainstalowanie dowolnego wirusa, lub bootblock'u znajdującego się w bibliotece.

BootX to godny polecenia program zwłaszcza, że wykonuje bardzo dużo opcji, które są w takich programach potrzebne, a poza tym jest on wykonany estetycznie i można go z pełną odpowiedzialnością nazwać user friendly.

Marcin "Duddie" Dudar

PUBLIC DOMAIN PACK

PUBLIC DOMAIN PACK C-64

Kwiecień

Strona A
- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku.

Strona B
- „ONE YEAR - RADIUS" - mega demo grupy RADIUS. Bardzo ładna grafika.

Maj

Strona A
- CRUEL SOLIDERS - demo
- DESTINATION'91 - demo
- SUCKER DJ! - demo (digi mix)
- MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów

Strona B
- MEGA DEMO „INFOSYSTEM 91"

Czerwiec

Strona A
- FONTEDITOR
- SINDATA EDITOR
- COLOR EDITOR
- DISK - NOTER
- GWIAZDY - demo graficzne
- FILGRAEPH 2.2/BML
- NOTE TO FLI V 2.2
- AFLI - EDITOR V 1.2
- RESET - MON,8,1
- TURBO - ASS 5
- ...HIGHLIFE #5
- AXEL NEWS #1
- DISK NOTKA/PADUA

Strona B
- PSC - MAG #9'06/91
- CONSPIRE/OREGON - demo
- CONTACT DEMO/ORE
- SHOWPIX

Lipiec

Strona A
- Mega demo „MY, OH MY!" grupy LIGHT

Strona B
- Game Music Composer - edytor muzyczny grupy GRAFFITY Węgier.

Sierpień

Strona A
- MegaDemo „Unnamed" grupy CAMELOT
- Sound Killer - edytor muzyczny grupy TOPAZ
- AFLI - Editor - edytor graficzny techniki A-FLI
- Disk-Dos obsługa komend stacji dysków
- Noter v2.2 grupy TOPAZ
- IFFL - Squeezer kompresor dyskowy
- Dirmaster+ - edytor do dyskietek
- Super Copy - DOS szybki program kopiujący do zbiorów

Strona B
- Mega Demo fińskiej grupy TOPAZ - „Graveyard Blues"

PUBLIC DOMAIN PACK AMIGA

Kwiecień

- RUBBER VECTORS - demo
- KEFTALES - demo
- DISK MASTER V3.0
- Moduły muzyczne:
  - TECHNOSTYLE 2
  - GALAXY 2
- GRAFIKA - prezentujemy rysunki
  - RICK PARKS

Maj

- VIRUS X 5.0
- VIRUS TERMINATOR
- PARADOX - demo
- STORMCHILD - demo
- Moduły muzyczne:
  - MIAMI VOICE
  - ANTI ATARI SONG

Czerwiec

- POWER BOOT - własne menu dysku
- DISK CODING SYSTEM - program do zabezpieczania dysków
- Konwerter IFF - ANSI
- AUER NATION - demo
- Moduły muzyczne
- DOCS - opis gry ELWIRA
- LAMER DEFENCE - do wykrywania i niszczenia wirusów
- REWENG GO OF THE LAMER - grafika w trybie D_HAM

Lipiec

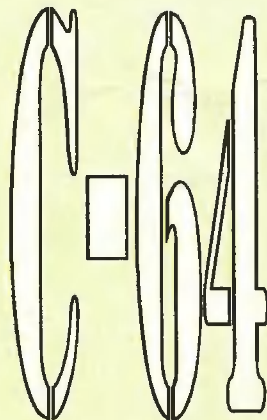
- Sanity - demo
- Amiga - Tanx (1Mb) - gra
- Little Beau (1MB) - gra
- There is A Light/Tonid - modules

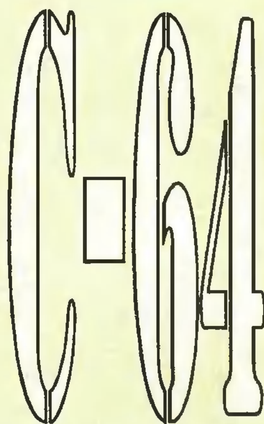
Spis treści zestawów z poprzednich miesięcy - patrz wcześniejsze numery naszego pisma.

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr:5.09011-400522.7-136-11-111.0 następujących kwot: 20.000zł za pojedyńczy zestaw dla C-64, 25.000zł za zestaw dla AMIGI. Kwoty te obejmują koszt dyskietki, koszty kopiowania, opakowania i przesyłki pocztowej. Blankiety wpłat powinny być **CZYTELNIE** wypełnione i zawierać: **imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64"**(jeśli zamawiamy zestaw dla C-64) lub **„PDP-A"**(zestaw dla Amigi) - dane te prosimy umieszczać na **wszystkich** odcinkach dowodu wpłaty. W prenumeracie zestawy kosztują: PDP-64 - 18.000zł (12 numerów 216tys zł), PDP-A - 22.000zł (12 numerów 264tys zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Prenumerata może obejmować miesiące od początku roku - tzn. zamawiając całoroczną prenumeratę np. w październiku, w pierwszej przesyłce otrzymacie wszystkie poprzednie zestawy.

ZAMÓW NIE ZWLEKAJ!

VOICETRACKER V4.0





Rewelacyjny program muzyczny!

Tylko **50.000 zł** kosztuje fantastyczny edytor muzyczny wykorzystujący ogromne możliwości dźwiękowe komputera Commodore - 64. Oferowany zestaw zawiera dyskietkę lub taśmę magnetofonową z programem VOICETRACKER V4.0, trzydzieści demonstracji muzycznych, oraz dokładną instrukcję. **UWAGA! Wersja magnetofonowa tylko 40.000zł.!**

Przedsiębiorstwo ABUK posiada wyłączność na dystrybucję tego programu. Wszelkie kopiowanie programu i powielanie instrukcji jest zabronione. Nabywcy otrzymują rejestrowane kopie programu wraz z prawem nabywania nowych wersji po znacznie obniżonych cenach oraz wymiany dyskietki w razie uszkodzenia. Studiom komputerowym proponujemy zakup hurtowy (przy zakupie powyżej 10 kompletów udzielamy 20% rabatu).

Chcąc stać się posiadaczem programu VOICETRACKER V4.0 wystarczy dokonać wpłaty 50.000zł (wersja dyskowa) lub 40.000zł (taśma) na konto: Bank PKO SA Bydgoszcz, konto nr: 5.09011-400522.7-136-11-111.0.

Na blankiecie prosimy czytelnie podać swoje imię, nazwisko i adres wraz z dopiskiem „VV4.0" uzupełnionym literką „T" - taśma lub „D" - dyskietka.

